

# **Processor Programming Reference (PPR) for AMD Family 1Ah Model 44h, Revision B0 Processors**

# Legal Notices

© 2023,2024 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.

Adobe is a registered trademark of Adobe.

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

DirectX is a registered trademark of Microsoft Corporation.

HDMI is a trademark of HDMI Licensing, LLC.

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

Linux is a registered trademark of Linus Torvalds.

Microsoft is a registered trademark of Microsoft Corporation.

OpenCL is a trademark of Apple, Inc. used by permission by Khronos Group, Inc.

OpenGL is a registered trademark of Hewlett Packard Enterprise.

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

PCIe is a registered trademark of PCI-SIG Corporation.

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

- 1 Overview**
- 2 Core Complex (CCX)**
- 3 Reliability, Availability, and Serviceability (RAS) Features**
- 4 Advanced Platform Management Link (APML)**
- 5 SB Temperature Sensor Interface (SB-TSI)**

**List of Namespaces**

**List of Definitions**

# Table of Contents

<b>1</b>	<b>Overview</b>
1.1	Intended Audience
1.2	Reference Documents
1.2.1	Documentation Conventions
1.3	Adobe® Reader
1.3.1	Adobe® Reader Configuration
1.3.1.1	Open Hyperlink Document in New Window
1.3.2	Adobe® Reader Usage
1.3.2.1	Searching a Multiple Volume PPR
1.3.2.2	Cross-References and Hyperlinks
1.3.2.3	Find Current Bookmark
1.4	Conventions
1.4.1	Numbering
1.4.2	Arithmetic And Logical Operators
1.4.2.1	Operator Precedence and Associativity
1.4.3	Register Mnemonics
1.4.3.1	Logical Mnemonic
1.4.3.2	Physical Mnemonic
1.4.4	Register Format
1.4.4.1	Register Instances
1.4.4.2	Register Physical Mnemonic, Title, and Name
1.4.4.3	Full Width Register Attributes
1.4.4.4	Register Description
1.4.4.5	Register Instance Table
1.4.4.5.1	Content Ordering in a Row
1.4.4.5.2	Multiple Instances Per Row
1.4.4.5.3	MSR Access Method
1.4.4.5.3.1	MSR Per-Thread Example
1.4.4.5.3.2	MSR Range Example
1.4.4.5.4	BAR Access Method
1.4.4.5.4.1	BAR as a Register Reference
1.4.4.5.5	PCICFG Access Method
1.4.4.5.5.1	PCICFG Bus Implied to be 00h
1.4.4.5.6	Data Port Access Method
1.4.4.6	Register Field Format
1.4.4.7	Simple Register Field Format
1.4.4.8	Complex Register Field Format
1.4.4.9	Field Name is Reserved
1.4.4.10	Field Access Type
1.4.4.10.1	Conditional Access Type Expression
1.4.4.11	Field Reset
1.4.4.12	Field Initialization
1.4.4.13	Field Check
1.4.4.14	Field Valid Values
1.4.5	Revision History and Change Bar Notation
1.5	Definitions
1.6	Changes Between Revisions and Product Variations
1.6.1	Revision Conventions
1.7	Package
1.7.1	Package type

- 1.8 Processor Overview
  - 1.8.1 Features
- 2 **Core Complex (CCX)**
  - 2.1 Processor x86 Core
    - 2.1.1 Core Functional Information
      - 2.1.1.1 Core Definitions
    - 2.1.2 Secure Virtual Machine Mode (SVM)
      - 2.1.2.1 BIOS support for SVM Disable
        - 2.1.2.1.1 Enable AMD Virtualization™
        - 2.1.2.1.2 Disable AMD Virtualization™
        - 2.1.2.1.3 Disable AMD Virtualization™, with a user supplied key
    - 2.1.3 Microcode Patching
      - 2.1.3.1 Microcode Patching Overview
      - 2.1.3.2 Microcode Patch Block (MPB)
    - 2.1.4 Effective Frequency
    - 2.1.5 Address Space
      - 2.1.5.1 Virtual Address Space
      - 2.1.5.2 Physical Address Space
      - 2.1.5.3 System Address Map
        - 2.1.5.3.1 Memory Access to the Physical Address Space
          - 2.1.5.3.1.1 Determining Memory Type
    - 2.1.6 Configuration Space
      - 2.1.6.1 Memory Mapped IO (MMIO) Configuration Coding Requirements
      - 2.1.6.2 MMIO Configuration Ordering
      - 2.1.6.3 Processor Configuration Space
    - 2.1.7 PCI Configuration Legacy Access
    - 2.1.8 System Software Interaction With SMT Enabled
    - 2.1.9 Register Sharing
    - 2.1.10 Timers
    - 2.1.11 Interrupts
      - 2.1.11.1 System Management Mode (SMM)
        - 2.1.11.1.1 SMM Overview
        - 2.1.11.1.2 Mode and Default Register Values
        - 2.1.11.1.3 SMI Sources And Delivery
        - 2.1.11.1.4 SMM Initial State
        - 2.1.11.1.5 SMM Save State
        - 2.1.11.1.6 System Management State
        - 2.1.11.1.7 Exceptions and Interrupts in SMM
        - 2.1.11.1.8 The Protected ASeg and TSeg Areas
        - 2.1.11.1.9 SMM Special Cycles
        - 2.1.11.1.10 Locking SMM
        - 2.1.11.1.11 SMM Page Configuration Lock
      - 2.1.11.2 Local APIC
        - 2.1.11.2.1 Local APIC Functional Description
          - 2.1.11.2.1.1 Detecting and Enabling
          - 2.1.11.2.1.2 APIC Register Space
          - 2.1.11.2.1.3 ApicId Enumeration Requirements
          - 2.1.11.2.1.4 Physical Destination Mode
          - 2.1.11.2.1.5 Logical Destination Mode
          - 2.1.11.2.1.6 Interrupt Delivery
          - 2.1.11.2.1.7 Vectored Interrupt Handling
          - 2.1.11.2.1.8 Interrupt Masking
          - 2.1.11.2.1.9 Spurious Interrupts

- 2.1.11.2.1.10 Spurious Interrupts Caused by Timer Tick Interrupt
- 2.1.11.2.1.11 Lowest-Priority Interrupt Arbitration
- 2.1.11.2.1.12 Inter-Processor Interrupts
- 2.1.11.2.1.13 APIC Timer Operation
- 2.1.11.2.1.14 Generalized Local Vector Table
- 2.1.11.2.1.15 State at Reset
- 2.1.11.2.2 Local APIC Registers
- 2.1.12 CUID Instruction
- 2.1.12.1 CUID Instruction Functions
- 2.1.13 MSR Registers
- 2.1.13.1 MSRs - MSR0000\_0xxx
- 2.1.13.2 MSRs - MSRC000\_0xxx
- 2.1.13.3 MSRs - MSRC001\_0xxx
- 2.1.13.4 MSRs - MSRC001\_1xxx
- 2.1.13.5 L3 MSRs - MSRC001\_1xxx
- 2.1.14 Performance Monitor Counters
- 2.1.14.1 RDPMC Assignments
- 2.1.14.2 Performance Measurement
- 2.1.14.3 Pipeline Utilization Analysis
- 2.1.14.4 Large Increment per Cycle Events
- 2.1.14.5 Core Performance Monitor Counters
- 2.1.14.5.1 Floating-Point (FP) Events
- 2.1.14.5.2 Load/Store (LS) Events
- 2.1.14.5.3 Instruction Cache (IC) and Branch Prediction (BP) Events
- 2.1.14.5.4 DE Events
- 2.1.14.5.5 EX (SC) Events
- 2.1.14.5.6 L2 Cache Events
- 2.1.14.6 L3 Cache Performance Monitor Counters
- 2.1.14.6.1 L3 Cache PMC Events
- 2.1.15 Instruction Based Sampling (IBS)
- 2.2 L3 Cache
- 2.2.1 L3 MSR Registers
- 2.2.2 L3 Clocks and Test (CT) MSR Registers.

### 3 Reliability, Availability, and Serviceability (RAS) Features

- 3.1 Machine Check Architecture
- 3.1.1 Overview
- 3.1.1.1 Legacy Machine Check Architecture
- 3.1.1.2 Machine Check Architecture Extensions
- 3.1.1.3 Use of MCA Information
- 3.1.1.3.1 Error Management
- 3.1.1.3.2 Fault Management
- 3.1.2 Machine Check Registers
- 3.1.2.1 Global Registers
- 3.1.2.2 Machine Check Banks
- 3.1.2.2.1 Legacy MCA Registers
- 3.1.2.2.2 Legacy MCA MSRs
- 3.1.2.2.3 MCAX Registers
- 3.1.2.2.4 MCAX MSRs
- 3.1.2.3 Access Permissions
- 3.1.3 Machine Check Errors
- 3.1.3.1 Error Severities
- 3.1.3.2 Exceptions and Interrupts
- 3.1.3.3 Error Codes

- 3.1.3.4 Extended Error Codes
- 3.1.3.5 DOER and SEER State
- 3.1.3.6 MCA Overflow Recovery
- 3.1.3.7 MCA Recovery
- 3.1.4 Machine Check Features
  - 3.1.4.1 Error Thresholding
  - 3.1.4.2 Error Simulation
- 3.1.5 Software Guidelines
  - 3.1.5.1 Recognizing MCAX Support
  - 3.1.5.2 Communicating MCAX Support
  - 3.1.5.3 Machine Check Initialization
  - 3.1.5.4 Determining Bank Count
  - 3.1.5.5 Determining Bank Type
  - 3.1.5.6 Recognizing Error Type
  - 3.1.5.7 Machine Check Error Handling
- 3.2 Machine Check Architecture Implementation
  - 3.2.1 Implemented Machine Check Banks
  - 3.2.2 Implemented Machine Check Bank Registers
  - 3.2.3 Mapping of Banks to Blocks
  - 3.2.4 Mapping of Banks to Blocks
  - 3.2.5 Decoding Error Type
  - 3.2.6 MCA Banks
    - 3.2.6.1 LS
    - 3.2.6.2 IF
    - 3.2.6.3 L2
    - 3.2.6.4 DE
    - 3.2.6.5 EX
    - 3.2.6.6 FP
    - 3.2.6.7 L3
    - 3.2.6.8 CS
    - 3.2.6.9 PIE
    - 3.2.6.10 UMC
    - 3.2.6.11 MP5
    - 3.2.6.12 NBIO
    - 3.2.6.13 KPX SERDES
    - 3.2.6.14 KPX GMI
    - 3.2.6.15 PCS GMI
- 4 Advanced Platform Management Link (APML)**
  - 4.1 Overview
    - 4.1.1 Definitions
  - 4.2 SBI Bus Characteristics
    - 4.2.1 SMBus Protocol Support
    - 4.2.2 I2C Support
  - 4.3 SBI Processor Information
    - 4.3.1 SBI Processor Pins
      - 4.3.1.1 Physical Layer Characteristics
    - 4.3.2 Processor States
  - 4.4 SBI Protocols
    - 4.4.1 SBI Modified Block Write-Block Read Process Call
    - 4.4.2 SBI Remote Management Interface (SB-RMI)
      - 4.4.2.1 SB-RMI Processor State Access
        - 4.4.2.1.1 SB-RMI Read Processor Register and Read CPUID Commands
        - 4.4.2.1.2 SB-RMI Write Processor Register Command

- 4.4.2.1.3 SB-RMI Protocol Status Codes
- 4.4.2.2 SB-RMI Mailbox Service
  - 4.4.2.2.1 SB-RMI Mailbox Sequence
- 4.4.2.3 SB-RMI Boot code status
- 4.4.2.4 SB-RMI Register Access
  - 4.4.2.4.1 SB-RMI Register Block Access
  - 4.4.2.4.2 SB-RMI Register Byte Access
- 4.4.2.5 SB-RMI Alert
- 4.4.3 SBI Error Detection and Recovery
  - 4.4.3.1 Error Detection
    - 4.4.3.1.1 ACK/NAK Mechanism
    - 4.4.3.1.2 Packet Error Correction (PEC)
    - 4.4.3.1.3 Bus Timeouts
  - 4.4.3.2 Error Recovery
    - 4.4.3.2.1 SBI Bus Reset
- 4.5 SBI Physical Interface
  - 4.5.1 SBI SMBus Address
  - 4.5.2 SBI Bus Timing
- 4.6 SB-RMI Registers
- 5 SB Temperature Sensor Interface (SB-TSI)**
  - 5.1 Overview
    - 5.1.1 Definitions
  - 5.2 SB-TSI Protocol
    - 5.2.1 SB-TSI Send/Receive Byte Protocol
      - 5.2.1.1 SB-TSI Address Pointer
    - 5.2.2 SB-TSI Read/Write Byte Protocol
    - 5.2.3 Alert Behavior
    - 5.2.4 Atomic Read Mechanism
    - 5.2.5 SB-TSI Temperature and Threshold Encodings
    - 5.2.6 SB-TSI Temperature Offset Encoding
  - 5.3 SB-TSI Physical Interface
    - 5.3.1 SB-TSI SMBus Address
    - 5.3.2 SB-TSI Bus Timing
    - 5.3.3 SB-TSI Bus Electrical Parameters
    - 5.3.4 Pass-FET Option
  - 5.4 SB-TSI Registers



# List of Figures

Figure 1:	Adobe® Reader Hyperlink Opens New Window Configuration
Figure 2:	Adobe® Reader Select Between Opened Files
Figure 3:	Adobe® Reader Searching a Multiple Volume PPR
Figure 4:	Adobe® Reader Find Current Bookmark Button
Figure 5:	Register Physical Mnemonic, Title, and Name
Figure 6:	Full Width Register Attributes
Figure 7:	Register Description
Figure 8:	Register Instance Table: Content Ordering in a Row
Figure 9:	Register Instance Table: MSR Example
Figure 10:	Register Instance Table: MSR Range Example
Figure 11:	Register Instance Table: BAR as Register Reference
Figure 12:	Register Instance Table: Bus Implied to be 00h
Figure 13:	Register Instance Table: Data Port Select
Figure 14:	Simple Register Field Example
Figure 15:	Register Field Sub-Row for {Reset,AccessType,Init,Check}
Figure 16:	Register Field Sub-Row for Description
Figure 17:	Register Field Sub-Row for Valid Value Table
Figure 18:	Register Field Sub-Row for Valid Bit Table
Figure 19:	Revision History Format Example
Figure 20:	Change Notation Example
Figure 21:	Overview of Family 1A Core Component Numbering
Figure 22:	Register Sharing Domains
Figure 23:	Instance Parameters
Figure 24:	SBI Transmission Protocol
Figure 25:	RTS Thermal Management Example
Figure 26:	SB-TSI Thermal Management Example
Figure 27:	Alert Assertion Diagram
Figure 28:	Pass FET Implementation

# List of Tables

Table 1:	Reference Documents Listing
Table 2:	Arithmetic and Logical Operator Definitions
Table 3:	Function Definitions
Table 4:	Operator Precedence and Associativity
Table 5:	Register Mnemonic Definitions
Table 6:	Logical Mnemonic Definitions
Table 7:	Physical Mnemonic Definitions
Table 8:	AccessType Definitions
Table 9:	Reset Type Definitions
Table 10:	Init Type Definitions
Table 11:	Definitions
Table 12:	Package Definitions
Table 13:	Definitions
Table 14:	MPB Definition
Table 15:	SMM Initial State
Table 16:	SMM Save State
Table 17:	ICR Valid Combinations
Table 18:	Guidance for Common Performance Statistics with Complex Event Selects
Table 19:	Guidance for Pipeline Utilization Analysis Statistics
Table 20:	PMC_Definitions
Table 21:	Machine Check Terms and Acronyms
Table 22:	Legacy MCA MSR Layout
Table 23:	MCAX MSR Layout
Table 24:	MCAX Implementation-Specific Register Layout
Table 25:	Error Overwrite Priorities
Table 26:	Error Scope Hierarchy
Table 27:	Error Code Types
Table 28:	Error code: transaction type (TT)
Table 29:	Error codes: cache level (LL)
Table 30:	Error codes: memory transaction type (RRRR)
Table 31:	Blocks Capable of Supporting MCA Banks
Table 32:	Mapping of Blocks to MCA_IPID[HwId] and MCA_IPID[McaType]
Table 33:	Legacy MCA Registers
Table 34:	MCAX Registers
Table 35:	Core MCA Bank to Block Mapping
Table 36:	Non-core MCA Bank to Block Mapping
Table 37:	MCA Bank to Block Mapping
Table 38:	MCA_STATUS_LS
Table 39:	MCA_ADDR_LS
Table 40:	MCA_SYND_LS
Table 41:	MCA_STATUS_IF
Table 42:	MCA_ADDR_IF
Table 43:	MCA_SYND_IF
Table 44:	MCA_STATUS_L2
Table 45:	MCA_ADDR_L2
Table 46:	MCA_SYND_L2
Table 47:	MCA_STATUS_DE
Table 48:	MCA_ADDR_DE
Table 49:	MCA_SYND_DE
Table 50:	MCA_STATUS_EX

Table 51:	MCA_ADDR_EX
Table 52:	MCA_SYND_EX
Table 53:	MCA_STATUS_FP
Table 54:	MCA_ADDR_FP
Table 55:	MCA_SYND_FP
Table 56:	MCA_STATUS_L3
Table 57:	MCA_ADDR_L3
Table 58:	MCA_SYND_L3
Table 59:	MCA_STATUS_CS
Table 60:	MCA_ADDR_CS
Table 61:	MCA_SYND_CS
Table 62:	MCA_STATUS_PIE
Table 63:	MCA_ADDR_PIE
Table 64:	MCA_SYND_PIE
Table 65:	MCA_STATUS_UMC
Table 66:	MCA_ADDR_UMC
Table 67:	MCA_SYND_UMC
Table 68:	MCA_STATUS_MP5
Table 69:	MCA_ADDR_MP5
Table 70:	MCA_SYND_MP5
Table 71:	MCA_STATUS_NBIO
Table 72:	MCA_ADDR_NBIO
Table 73:	MCA_SYND_NBIO
Table 74:	MCA_STATUS_KPX_SERDES
Table 75:	MCA_ADDR_KPX_SERDES
Table 76:	MCA_SYND_KPX_SERDES
Table 77:	MCA_STATUS_KPX_GMI
Table 78:	MCA_ADDR_KPX_GMI
Table 79:	MCA_SYND_KPX_GMI
Table 80:	MCA_STATUS_PCS_GMI
Table 81:	MCA_ADDR_PCS_GMI
Table 82:	MCA_SYND_PCS_GMI
Table 83:	APML Definitions
Table 84:	SB-RMI Functions
Table 85:	SB-RMI Read Processor Register Command Protocol
Table 86:	SB-RMI Read CPUID Command Protocol
Table 87:	SB-RMI Read Data/Status Command Protocol
Table 88:	SB-RMI Load Address Command Protocol
Table 89:	SB-RMI Write Processor Register Command Protocol
Table 90:	SB-RMI Status Codes
Table 91:	SB-RMI Soft Mailbox Message
Table 92:	SB-RMI Soft Mailbox Error Code
Table 93:	SB-RMI Register Block Write Protocol
Table 94:	SB-RMI Register Block Read Protocol
Table 95:	SB-RMI Register Write Byte Protocol
Table 96:	SB-RMI Register Read Byte Protocol
Table 97:	SB-TSI Definitions
Table 98:	SB-TSI CPU Temperature and Threshold Encoding Examples
Table 99:	SB-TSI Temperature Offset Encoding Examples
Table 100:	SB-TSI Address Encodings

## 1 Overview

### 1.1 Intended Audience

This document provides the processor behavioral definition and associated design notes. It is intended for platform designers and for programmers involved in the development of BIOS functions, drivers, and operating system kernel modules.

### 1.2 Reference Documents

*Table 1: Reference Documents Listing*

Term	Description
<b>docAPM1</b>	AMD64 Architecture Programmer's Manual Volume 1: Application Programming, order# 24592.
<b>docAPM2</b>	AMD64 Architecture Programmer's Manual Volume 2: System Programming, order# 24593.
<b>docAPM3</b>	AMD64 Architecture Programmer's Manual Volume 3: Instruction-Set Reference, order# 24594.
<b>docAPM4</b>	AMD64 Architecture Programmer's Manual Volume 4: 128-Bit and 256-Bit Media Instructions, order# 26568.
<b>docAPM5</b>	AMD64 Architecture Programmer's Manual Volume 5: 64-Bit Media and x87 Floating-Point Instructions, order# 26569.
<b>docACPI</b>	Advanced Configuration and Power Interface (ACPI) Specification. <a href="http://uefi.org/specifications">http://uefi.org/specifications</a> .
<b>docASF</b>	Alert Standard Format Specification. <a href="http://dmtf.org/standards/asf">http://dmtf.org/standards/asf</a> .
<b>docATA</b>	AT Attachment with Packet Interface. <a href="http://www.t13.org">http://www.t13.org</a> .
<b>docDP</b>	VESA DisplayPort Standard. <a href="http://www.vesa.org/vesa-standards">http://www.vesa.org/vesa-standards</a> .
<b>docIOMMU</b>	AMD I/O Virtualization Technology Specification, order# 48882.
<b>docI2C</b>	I2C Bus Specification. <a href="http://www.nxp.com/documents/user_manual/UM10204.pdf">http://www.nxp.com/documents/user_manual/UM10204.pdf</a>
<b>docJEDEC</b>	JEDEC Standards. <a href="http://www.jedec.org">http://www.jedec.org</a> .
<b>docPCIe</b>	PCI Express® Specification. <a href="http://www.pcisig.org">http://www.pcisig.org</a> .
<b>docPCIb</b>	PCI Local Bus Specification. <a href="http://www.pcisig.org">http://www.pcisig.org</a> .
<b>docSATA</b>	Serial ATA Specification. <a href="http://www.sata-io.org">http://www.sata-io.org</a> .
<b>docSDHC</b>	Secure Digital Host Controller Standard Specification. <a href="https://www.sdcard.org">https://www.sdcard.org</a> .
<b>docUSB</b>	Universal Serial Bus Specification. <a href="http://www.usb.org">http://www.usb.org</a> .

#### 1.2.1 Documentation Conventions

When referencing information found in external documents listed in Reference Documents, the "=>" operator is used. This notation represents the item to be searched for in the reference document. For example:

docExDoc => Header1 => Header2

is to have the reader use the search facility when opening referenced document "docExDoc" and search for "Header2". "Header2" may appear more than once in "docExDoc", therefore, referencing the one that follows "Header1". In that case, the easiest way to get to Header2 is to use the search to locate Header1, then again to locate "Header2".

### 1.3 Adobe® Reader

This section describes how to configure and use Adobe® Reader for the PPR PDFs.

Adobe Reader is the recommended tool for viewing PPR pdfs and can be downloaded at <https://get.adobe.com/reader/>.

### 1.3.1 Adobe® Reader Configuration

This section describes how to configure Adobe Reader for the PPR PDFs.

#### 1.3.1.1 Open Hyperlink Document in New Window

The Open Hyperlink Document in New Window setting opens a new window for a hyperlink, instead of opening the hyperlink document in the same window.

Menu->Preferences:

- Documents
  - Open Settings:
    - Deselect: Open cross-document links in same window

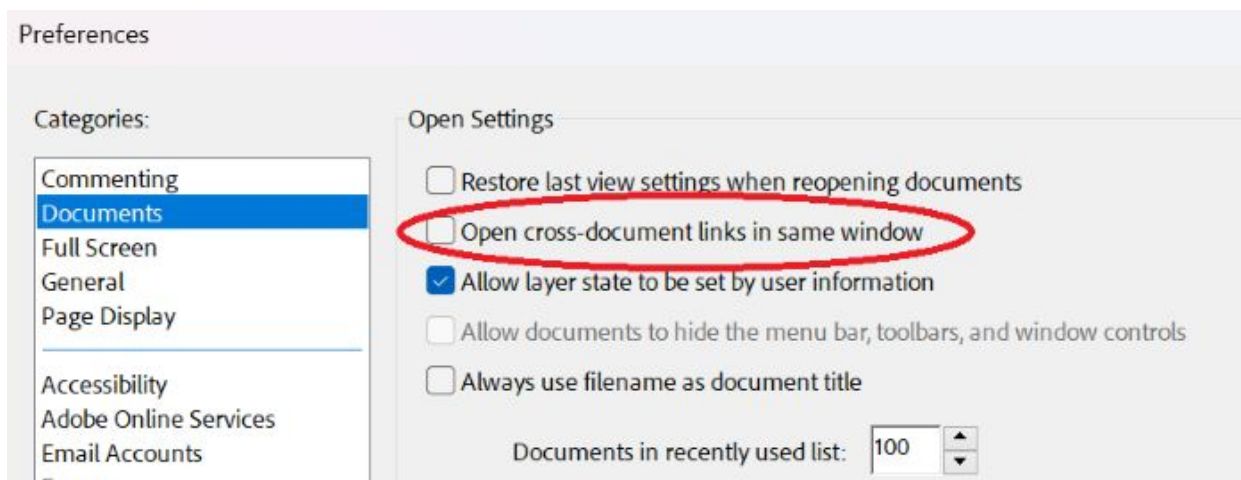


Figure 1: Adobe® Reader Hyperlink Opens New Window Configuration

The following Figure shows how when hyperlinking from one document to another, the original document is left open. The tab that is not grayed-out indicates the foreground window.

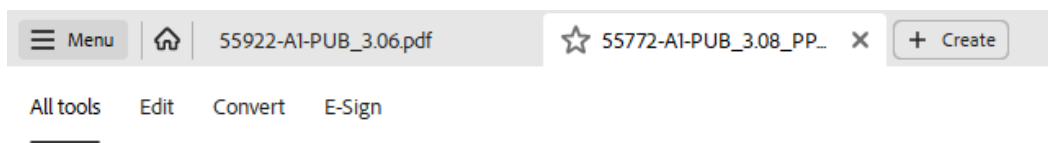


Figure 2: Adobe® Reader Select Between Opened Files

### 1.3.2 Adobe® Reader Usage

This section describes how to use Adobe Reader for the PPR PDFs.

NOTE: PDF's are distributed in zip format. In order to search and hyperlink between PDF volumes, the zip contents must

be extracted to a folder.

### 1.3.2.1 Searching a Multiple Volume PPR

The PPR is a multiple PDF document and searching all PDFs is performed as follows:

- The zip of PDF files must be extracted to a directory where the search will be performed. A search across multiple PDF files can not be performed from within a zip of PDF's.
- Open search by selecting Menu->Search->Advanced Search (Shift+Ctrl+F)
- Select "All PDF Documents in" and select "Browse for Location...", which opens the "Browse For Folder" window.
- In the "Browse For Folder" window, select the folder that contains the PPR PDFs that need to be searched, and select OK.

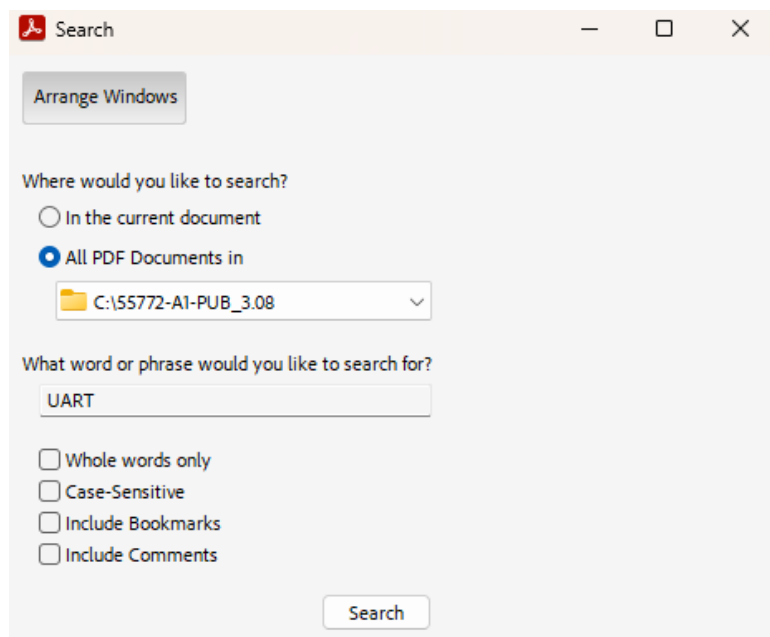


Figure 3: Adobe® Reader Searching a Multiple Volume PPR

### 1.3.2.2 Cross-References and Hyperlinks

A cross-reference is a link to a location within the same PDF. A hyperlink is a link to a location within a different PDF.

- For cross-references, use "Previous View" to return from the current location to the previous location.
  - Menu->View->Page Navigation->Previous View
- Hyperlinks between documents leave the current location unchanged in the PDF that contained the hyperlink.
- In order for hyperlinks to work properly the zip of PDF's must be extracted to a directory. Hyperlinks will not function within a zip of PDF's.

### 1.3.2.3 Find Current Bookmark

The bookmark pane can highlight the current bookmark associated with the viewer pane by selecting the "find current bookmark" button, as shown below.

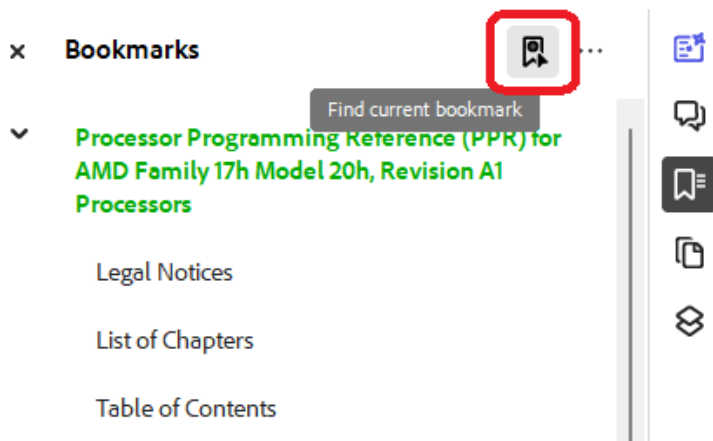


Figure 4: Adobe® Reader Find Current Bookmark Button

## 1.4 Conventions

### 1.4.1 Numbering

- Binary numbers: Binary numbers are indicated either by appending a "b" at the end (e.g., 0110b) or by Verilog syntax (e.g., 4'b0110).
- Hexadecimal numbers: Hexadecimal numbers are indicated by appending an "h" to the end (e.g., 45F8h) or by Verilog syntax (e.g., 16'h45F8).
- Decimal numbers: A number is decimal if not specified to be binary or hex.
- Exception: Physical register mnemonics are implied to be hex without the h suffix.
- Underscores in numbers: Underscores are used to break up numbers to make them more readable. They do not imply any operation (e.g., 0110\_1100).

### 1.4.2 Arithmetic And Logical Operators

In this document, formulas generally follow Verilog conventions for logic equations.

Table 2: Arithmetic and Logical Operator Definitions

Operator	Definition
{}	Concatenation. Curly brackets are used to indicate a group of bits that are concatenated together. Each set of bits is separated by a comma (e.g., {Addr[3:2], Xlate[3:0]} represents a 6-bit values; the two MSBs are Addr[3:2] and the four LSBs are Xlate[3:0]).
	Bitwise OR (e.g., 01b   10b == 11b).
	Logical OR (e.g., 01b    10b == 1b). It treats a multi-bit operand as 1 if >= 1 and produces a 1-bit result.
&	Bitwise AND (e.g., 01b & 10b == 00b).
&&	Logical AND (e.g., 01b && 10b == 1b). It treats a multi-bit operand as 1 if >= 1 and produces a 1-bit result.
^	Bitwise exclusive-OR (e.g., 01b ^ 10b == 11b). Sometimes used as "raised to the power of" as well, as indicated by the context in which it is used (e.g., 2^2 == 4).
~	Bitwise NOT (also known as one's complement). (e.g., ~10b == 01b).
!	Logical NOT (e.g., !10b == 0b). It treats a multi-bit operand as 1 if >= 1 and produces a 1-bit result.

<, <=, >, >=, ==, !=	Relational. Less than, Less than or equal, greater, greater than or equal, equal, and not equal.
+, -, *, /, %	Arithmetic. Addition, subtraction, multiplication, division, and modulus.
<<	Bitwise left shift. Shift left first operand by the number of bits specified by the 2nd operand (e.g., 01b << 01b == 10b).
>>	Bitwise right shift. Shift right first operand by the number of bits specified by the 2nd operand (e.g., 10b >> 01b == 01b).
?:	Ternary conditional (e.g., condition ? value if true : value if false).

Table 3: Function Definitions

Term	Description
<b>ABS</b>	ABS(integer expression): Remove sign from signed value.
<b>FLOOR</b>	FLOOR(integer expression): Rounds real number down to nearest integer.
<b>CEIL</b>	CEIL(real expression): Rounds real number up to nearest integer.
<b>MIN</b>	MIN(integer expression list): Picks minimum integer or real value of comma separated list.
<b>MAX</b>	MAX(integer expression list): Picks maximum integer or real value of comma separated list.
<b>COUNT</b>	COUNT(integer expression): Returns the number of binary 1's in the integer.
<b>ROUND</b>	ROUND(real expression): Rounds to the nearest integer; halfway rounds away from zero.
<b>UNIT</b>	UNIT(register field reference): Input operand is a register field reference that contains a valid values table that defines a value with a unit (e.g., clocks, ns, ms, etc.). This function takes the value in the register field and returns the value associated with the unit (e.g., If the field had a valid value definition where 1010b was defined as 5 ns). Then if the field had the value of 1010b, then UNIT() would return the value 5.
<b>POW</b>	POW(base, exponent): POW(x,y) returns the value x to the power of y.

#### 1.4.2.1 Operator Precedence and Associativity

This document follows C operator precedence and associativity. The following table lists operator precedence (highest to lowest). Their associativity indicates in what order operators of equal precedence in an expression are applied. Parentheses are also used to group subexpressions to force a different precedence; such parenthetical expressions can be nested and are evaluated from inner to outer (e.g., "X = A || !B && C" is the same as "X = A || ((!B) && C)").

Table 4: Operator Precedence and Associativity

Operator	Description	Associativity
!, ~	Logical negation/bitwise complement	right to left
*, /, %	Multiplication/division/modulus	left to right
+, -	Addition/subtraction	left to right
<<, >>	Bitwise shift left, Bitwise shift right	left to right
<, <=, >, >=, ==, !=	Relational operators	left to right
&	Bitwise AND	left to right
^	Bitwise exclusive OR	left to right
	Bitwise inclusive OR	left to right
&&	Logical AND	left to right
	Logical OR	left to right
?:	Ternary conditional	right to left



### 1.4.3 Register Mnemonics

A register mnemonic is a short name that uniquely refers to a register, either all instances of that register, some instances, or a single instance.

Every register instance can be expressed in 2 forms, logical and physical, as defined below.

*Table 5: Register Mnemonic Definitions*

Term	Description
<b>logical mnemonic</b>	The register mnemonic format that describes the register functionally, what namespace to which the register belongs, a name for the register that connotes its function, and optionally, named parameters that indicate the different function of each instance (e.g., Link::Phy::PciDevVendIDF3). See 1.4.3.1 [Logical Mnemonic].
<b>physical mnemonic</b>	The register mnemonic that is formed based on the physical address used to access the register (e.g., D18F3x00). See 1.4.3.2 [Physical Mnemonic].

#### 1.4.3.1 Logical Mnemonic

The logical mnemonic format consists of a register namespace, a register name, and optionally a register instance specifier (e.g., register namespace::register name register instance specifier).

For Unb::PciDevVendIDF3:

- The register namespace is Unb, which is the UNB IP register namespace.
- The register name is PciDevVendIDF3, which reads as PCICFG device and vendor ID in Function 3.
- There is no register instance specifier because there is just a single instance of this register.

For Dct::Phy::CalMisc2\_dct[1:0]\_chiplet[BCST,3:0]\_pad[BCST,11:0]:

- The register namespace is Dct::Phy, which is the DCT PHY register namespace.
- The register name is CalMisc2, which reads as miscellaneous calibration register 2.
- The register instance specifier is \_dct[1:0]\_chiplet[BCST,3:0]\_pad[BCST,11:0], which indicates that there are 2 DCTPHY instances, each IP for this register has 5 chiplets (0-3 and BCST), and for each chiplet 13 pads (0-11 and BCST). This register has 130 instances. (2\*5\*13)

Table 6: Logical Mnemonic Definitions

Term	Description
<b>register namespace</b>	A namespace for which the register name must be unique. A register namespace indicates to which IP it belongs and an IP may have multiple namespaces. A namespace is a string that supports a list of "::" separated names. The convention is for the list of names to be hierarchical, with the most significant name first and the least significant name last (e.g., Link::Phy::Rx is the RX component in the Link PHY).
<b>register name</b>	A name that connotes the function of the register.
<b>register instance specifier</b>	The register instance specifier exists when there is more than one instance for a register. The register instance specifier consists of one or more register instance parameter specifier (e.g., The register instance specifier <code>_dct[1:0]_chiplet[BCST,3:0]_pad[BCST,11:0]</code> consists of 3 register instance parameter specifiers, <code>_dct[1:0]</code> , <code>_chiplet[BCST,3:0]</code> , and <code>_pad[BCST,11:0]</code> ).
<b>register instance parameter specifier</b>	A register instance parameter specifier is of the form <code>_register parameter name[register parameter value list]</code> (e.g., The register instance parameter specifier <code>_dct[1:0]</code> has a register parameter name of <code>dct</code> (The DCT PHY instance name) and a register parameter value list of "1:0" or 2 instances of DCT PHY).
<b>register parameter name</b>	A register parameter name is the name of the number of instances at some level of the logical hierarchy (e.g., The register parameter name <code>dct</code> specifies how many instances of the DCT PHY exist).
<b>register parameter value list</b>	The register parameter value list is the logical name for each instance of the register parameter name (e.g., For <code>_dct[1:0]</code> , there are 2 DCT PHY instances, with the logical names 0 and 1, but it should be noted that the logical names 0 and 1 can correspond to physical values other than 0 and 1). It is the purpose of the <code>AddressMappingTable</code> to map these register parameter values to physical address values for the register.

#### 1.4.3.2 Physical Mnemonic

The physical register mnemonic format varies by the access method. The following table describes the supported physical register mnemonic formats.

Table 7: Physical Mnemonic Definitions

Term	Description
<b>PCICFG</b>	The PCICFG, or PCI defined configuration space, physical register mnemonic format is of the form DXFYxZZZ. Bus 0 is implied, X specifies the hexadecimal device number (this may be 1 or 2 digits). Y specifies the function number. ZZZ specifies the hexadecimal byte address (this may be 2 or 3 digits). Example: D18F2x40 specifies the register at bus 0, device 18h, function 2, and address 40h. If the mnemonic starts with B, then the physical mnemonic format is BWDXFYxZZZ where WW specifies the hexadecimal bus number (1 or 2 hex digits) or "XX" implying that the bus is relocatable. Example; BXXD00F6x40 specifies that the bus is relocatable, B0AD00F2x000 specifies that the bus is 0Ah.
<b>BAR</b>	The BAR, or base address register, physical register mnemonic format is of the form PREFIXxZZZ. PREFIX is an all capital letter name that connotes the BAR to which the offset is added to get the physical address of the operation. ZZZ is the offset.
<b>MSR</b>	The MSR, or x86 model specific register, physical register mnemonic format is of the form MSRXXXX_XXXX, where XXXX_XXXX is the hexadecimal MSR number. This space is accessed through x86 defined RDMSR and WRMSR instructions.
<b>PMC</b>	The PMC, or x86 performance monitor counter, physical register mnemonic format is any of the forms {PMCxXXX, L2IPMCxXXX, NBPMCxXXX}, where XXX is the performance monitor select.
<b>CPUID</b>	The CPUID, or x86 processor identification state, physical register mnemonic format is of the form CPUID FnXXXX_XXXX_EiX[_xYYY], where XXXX_XXXX is the hex value in the EAX and YYY is the hex value in ECX.

#### 1.4.4 Register Format

A register is a group of register instances that have the same field format (same bit indices and field names).

##### 1.4.4.1 Register Instances

All instances of a register:

- Have the same:
  - Field bit indices and names
  - Field titles, descriptions, valid values.
  - Register title
  - Register description
- Fields may have different: (instance specific)
  - Access Type. See 1.4.4.10 [Field Access Type].
  - Reset. See 1.4.4.11 [Field Reset].
  - Init. See 1.4.4.12 [Field Initialization].
  - Check. See 1.4.4.13 [Field Check].

##### 1.4.4.2 Register Physical Mnemonic, Title, and Name

A register definition is identified by a table that starts with a heavy bold line. The information above the bold line in order is:

1. The physical mnemonic of the register.
  - A register that has multiple instances, may have instances that have different access methods, each with it's own physical mnemonic format.
  - In the event that there are multiple physical mnemonic formats, the physical mnemonic format chosen is the most commonly used physical mnemonic.

- The physical mnemonic is not intended to represent the physical mnemonics of all instances of the register. It is only a visual aid to identify a register when scanning down a list, for readers that prefer to find registers by physical mnemonic. If "..." occurs in the physical mnemonic, the range is first ... last. There is no implication as to how many instances exist between first and last. See 1.4.4.5 [Register Instance Table].
- 2. The register title in brackets.
- 3. The register name in parenthesis.

Physical Mnemonic	Title	Name
MSR0000_0010	[Time Stamp Counter]	(TSC)
Read-write, Volatile. Reset: 0000_0000_0000_0000h.		
Core::X86::Msr::TSC_three[1:0]_core[3:0]_thread[1:0]; MSR00000010		
Bits	Description	
63:0	TSC: <b>time stamp counter</b> . Read-write, Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the TSC P0 frequency based value (as if TSCRatio == 1.0) when (TSCRatio != 1.0).	

Figure 5: Register Physical Mnemonic, Title, and Name

#### 1.4.4.3 Full Width Register Attributes

The first line that follows the bold line contains the attributes that apply to all fields of the register. This row is rendered as a convenience to the reader and replicates content that exists in the register field.

- AccessType: If all non-reserved fields of a register have the same access type, then the access type is rendered in this row.
  - The supported access types are specified by 1.4.4.10 [Field Access Type].
  - The example figure shows that the access type "Read-write, Volatile" applies to all non-reserved fields of the register.
- Reset: If all non-reserved fields of a register have a constant reset and are all the same type (Warm, Cold, Fixed), then the full width register reset is rendered in this row. The example figure shows the reset "0000\_0000\_0000\_0000h". See 1.4.4.11 [Field Reset].
  - The value zero (0) is assumed for display purposes for all reserved fields.
- If none of the above content is rendered, then this row of the register is not rendered.

#### MSR0000\_0010 [Time Stamp Counter] (TSC)

Read-write, Volatile. Reset: 0000_0000_0000_0000h.	
Core::X86::Msr::TSC_three[1:0]_core[3:0]_thread[1:0]; MSR00000010	
Bits	Description
63:0	TSC: <b>time stamp counter</b> . Read-write, Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the TSC P0 frequency based value (as if TSCRatio == 1.0) when (TSCRatio != 1.0).

Figure 6: Full Width Register Attributes

#### 1.4.4.4 Register Description

The register description is optional and appears after the "full width register attributes" row and before the "register

instance table" rows. The register description can be one or more paragraphs.

#### PciDevVendIDF3 [Device/Vendor ID]

Read-only. Reset: 0000_1022h.	
A register description. That can be multiple paragraphs.	
Link::Phy::Tx::PciDevVendIDF3; D18F3x00	
Bits	Description
31:16	DeviceID: device ID. Read-only. Reset: Fixed,0000h.
15:0	VendorID: vendor ID. Read-only. Reset: Fixed,1022h. Init: 1234h.

Figure 7: Register Description

### 1.4.4.5 Register Instance Table

The zero or more rows of 8-pt font before the Bits/Description row is the register instance table.

The register instance table can generally be described as follows:

- Each row describes the access method of one or more register instances.
- If a row describes two or more instances, then the logical instance range, left to right, corresponds to the physical range, left to right.
- The absence of register instance rows indicates that the register exists for documentation purposes, and no access method is described for the register.

Because there are multiple access methods for all the registers, each of the following subsections describes an aspect of the register instance table in isolation.

#### 1.4.4.5.1 Content Ordering in a Row

Content in a register instance table row is ordered as follows:

- The text up to the first semicolon is the logical mnemonic.
  - See 1.4.3.1 [Logical Mnemonic].
- The text after the first semicolon is the physical mnemonic.
  - See 1.4.3.2 [Physical Mnemonic].
- Optionally, content after the physical mnemonic provides additional information about the access method for the register instances in the row.

#### BXXD00F0x000 (NB\_VENDOR\_ID)

Read-only. Reset: 1022h.	
Vendor ID Register	
IOHC::NB_VENDOR_ID_aliasHOST; BXXD00F0x000; BXX=IOHC::NB_BUS_NUM_CNTL_aliasSMN[NB_BUS_NUM]	
IOHC::NB_VENDOR_ID_aliasSMN; NBCFGx00000000; NBCFG=13B0_0000h	

Figure 8: Register Instance Table: Content Ordering in a Row

### 1.4.4.5.2 Multiple Instances Per Row

Multiple instances in a row is represented by a single dimension "range" in the logical mnemonic and the physical mnemonic.

The single dimension order of instances is the same for both the logical and physical mnemonic. The first logical mnemonic is associated with the first physical mnemonic, so forth for the 2nd, up until the last.

- Brackets indicates a list, most significant to least significant.
- The ":" character indicates a continuous range between 2 values.
- The "," character separates non-contiguous values.
- There are some cases where more than one logical mnemonic maps to a single physical mnemonic.

Note that it is implied that the MSR {lthree,core,thread} parameters are not part of a range.

Example:

NAMESP::REGNAME\_inst[BLOCK[5:0],BCST]\_aliasHOST; FFF1x00000088\_x[000[B:6]\_0001,00000000]

- There are 7 instances.
- NAMESP is the namespace.
- 6 instances are represented by the sub-range 000[B:6]\_0001.
- \_instBCST corresponds to FFF1x00000088\_x00000000.
- \_inst BLOCK 0 corresponds to FFF1x00000088\_x00060001.
- ...
- \_inst BLOCK 5 corresponds to FFF1x00000088\_x000B0001.

#### 1.4.4.5.3 MSR Access Method

The MSR parameters {lthree,core,thread} are implied by the identity of the core on which the RDMSR/WRMSR is being executed, and therefore are not represented in the physical mnemonic.

MSRs that are:

- per-thread have the {lthree,core,thread} parameters.
- per-core do not have the thread parameter.
- per-L3 do not have the {core,thread} parameters.
- common to all L3's do not have the {lthree,core,thread} parameters.

##### 1.4.4.5.3.1 MSR Per-Thread Example

An MSR that is per-thread has all three {lthree,core,thread} parameters and all instances have the same physical mnemonic.

#### MSR0000\_0010 [Time Stamp Counter] (TSC)

Read-write, Volatile. Reset: 0000_0000_0000_0000h.	
Core::X86::Msr::TSC_lthree[1:0]_core[3:0]_thread[1:0]; MSR00000010	
Bits	Description
63:0	<b>TSC: time stamp counter.</b> Read-write, Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the TSC P0 frequency based value (as if TSCRatio == 1.0) when (TSCRatio != 1.0).

Figure 9: Register Instance Table: MSR Example

##### 1.4.4.5.3.2 MSR Range Example

An MSR can exist as a range for a parameter other than the {lthree,core,thread} parameters.

In the following example the n parameter is a range. The \_n0 value corresponds to MSR0000\_0201, and so on.

**MSR0000\_0201 [Variable-Size MTRRs Mask] (MtrrVarMask)**

Reset: 0000_0000_0000_0000h.
Core::X86::Msr::MtrrVarMask [n[7:0]] lthree[1:0]_core[3:0]; MSR0000_0201[[F,D,B,9,7,5,3,1]]

Figure 10: Register Instance Table: MSR Range Example

**1.4.4.5.4 BAR Access Method**

The BAR access method is indicated by a physical mnemonic that has the form PREFIXxNUMBER.

- BAR, which stands for "Base Address Register", is the base address for the IP block and can be a constant, a register field, or an expression that consists of one or more register fields.
- Example: APICx0000. The BAR prefix is "APIC".

The BAR prefix represents either a constant or an expression that consists of a register reference.

**1.4.4.5.4.1 BAR as a Register Reference**

A relocatable BAR is when the base of an IP is not a constant.

- The prefix NTBPRIBAR0 represents the base of the IP, the value of which comes from the register NBIFEPFNCFG::BASE\_ADDR\_1\_aliasHOST\_instNBIF0\_func1[BASE\_ADDR].
- The address of the register is NBIFEPFNCFG::BASE\_ADDR\_1\_aliasHOST\_instNBIF0\_func1[BASE\_ADDR] + 00000h.

**NTBPRIBAR0x00000 (NTB\_SMU\_PCTRL0)**

Reset: 0000_0000h.
NTB::NTB_SMU_PCTRL0_aliasHOSTPRI; NTBPRIBAR0x00000;
NTBPRIBAR0=NBIFEPFNCFG::BASE_ADDR_1_aliasHOST_instNBIF0_func1[BASE_ADDR]
NTB::NTB_SMU_PCTRL0_aliasHOSTSEC; NTBSECBAR0x00000;
NTBSECBAR0=NBIFEPFNCFG::BASE_ADDR_1_aliasHOST_instNBIF2_func1[BASE_ADDR]
NTB::NTB_SMU_PCTRL0_aliasSMN; NTBx00000000; NTB=0400_0000h

Figure 11: Register Instance Table: BAR as Register Reference

**1.4.4.5.5 PCICFG Access Method**

The PCICFG access method is indicated by a physical mnemonic that has the form DXXFXxNUMBER. There are 2 cases:

- Bus omitted and implied to be 00h.
- Bus represented as BXX and indicates that the bus is indicated by a register field.

Example:

- Example: D18F0x000. (The bus, when omitted, is implied to be 00h)
- Example: BXXD0F0x000. (The bus as an expression that includes a register reference)

**1.4.4.5.5.1 PCICFG Bus Implied to be 00h**

Example:

- The absence of a B before the D14 implies that the bus is 0.



FCH::ITF::LPC::PciDevVendID_aliasHOST; D14F3x000
--

Figure 12: Register Instance Table: Bus Implied to be 00h

#### 1.4.4.5.6 Data Port Access Method

A data port requires that the data port select be written before the register is accessed via the data port.

- The registers behind a data port is also called an indirect address space.
- The implied access method is to first write the data port select and then to read/write the register at the data port address.
- There are some cases where there are 2 or more data port selects.

Example:

- The data port select value follows the "\_x".
- The data port select register follows the "DataPortWrite=".
- The access method for \_instPIE0\_aliasHOST is:
  1. Write 0005\_0001h to DF::FabricConfigAccessControl.
  2. Read/Write the PCICFG address D18F0x040.

DF::FabricBlockInstanceCount_inst[PIE0,BCST]_aliasHOST; D18F0x040_x[00050001,00000000]; DataPortWrite=DF::FabricConfigAccessControl
DF::FabricBlockInstanceCount_inst[PIE0,BCST]_aliasSMN; DFF0x00000040_x[00050001,00000000]; DFF0=0001_C000h; DataPortWrite=DF::FabricConfigAccessControl

Figure 13: Register Instance Table: Data Port Select

#### 1.4.4.6 Register Field Format

The register field definition are all rows that follow the Bits/Description row. Each field row represents the definition of a bit range, with the bit ranges ordered from most to least significant. There are 2 columns, with the left column defining the field bit range, and the right column containing the field definition.

There are 2 field definition formats, simple and complex. If the description can be described in the simple one paragraph format then the simple format is used, else the complex format is used.

#### 1.4.4.7 Simple Register Field Format

The simple register format compresses all content into a single paragraph with the following implied order:

1. Field Name (required)
  - Allowed to be Reserved. See 1.4.4.9 [Field Name is Reserved].
  - "FFXSE" in the example figure.
2. Field Title
  - "fast FXSAVE/FRSTOR enable" in the example figure.
3. Field Access Type. See 1.4.4.10 [Field Access Type].
  - In the example figure the access type is "Read-write".
4. Field Reset. See 1.4.4.11 [Field Reset].
  - In the example figure the reset is warm reset and "0".
5. Field Init. See 1.4.4.12 [Field Initialization].
6. Field Check. See 1.4.4.13 [Field Check].
7. Field Valid Values. If the valid values are single bit (e.g., 0=, 1=). See 1.4.4.14 [Field Valid Values].
  - In the example figure the 1= definition begins with "Enables" and ends with "mechanism".



- In the example figure there is no 0= definition.
8. Field Description. If it is a single paragraph.
- In the example figure the field description begins with "This is" and ends with "afterwards".

All fields that do not exist are omitted.

14	<b>FFXSE: fast FXSAVE/FRSTOR enable.</b> Read-write. Reset: 0. 1=Enables the fast FXSAVE/FRSTOR mechanism. A 64-bit operating system may enable the fast FXSAVE/FRSTOR mechanism if (Core::X86::Cpuid::FeatureExtIdEdx[FFXSR] == 1). This bit is set once by the operating system and its value is not changed afterwards.
----	--

Figure 14: Simple Register Field Example

#### 1.4.4.8 Complex Register Field Format

Content that cannot be expressed in the single paragraph format is broken out to a separate sub-row (a definition column row).

Additional sub-rows are added in the following order:

1. Complex expression for {Reset,AccessType,Init,Check}.
2. Instance specific {Reset,AccessType,Init,Check} values.
3. Description, if more than 1 paragraph.
4. Valid values, if more than 0=/1=. Or a Valid bit table. (see figure)

The following figure highlights a complex access type specification.

63:0	<b>APerfReadOnly: read-only actual core clocks counter.</b> Reset: 0. This register increments in proportion to the actual number of core clocks cycles while the core is in C0. See Core::X86::Msr::MPerfReadOnly. This register is not affected by writes to Core::X86::Msr::APERF.
	AccessType: Core::X86::Msr::HWCR[EffFreqReadOnlyLock] ? Read-only, Volatile : Read-write, Volatile.

Figure 15: Register Field Sub-Row for {Reset,AccessType,Init,Check}

The following figure highlights a complex description specification.

4	<b>INVDWBINVD: INVD to WBINVD conversion.</b> Read-write. Reset: 1. Check: 1. 1=Convert INVD to WBINVD.
	<b>Description:</b> This bit is required to be set for normal operation when any of the following are true: <ul style="list-style-type: none"> <li>• An L2 is shared by multiple threads.</li> <li>• An L3 is shared by multiple cores.</li> <li>• CC6 is enabled.</li> <li>• Probe filter is enabled.</li> </ul>

Figure 16: Register Field Sub-Row for Description

The following figure highlights a complex valid value table, used either when the field is more than 1 bit or when the definition is more than a single sentence.

2:1	<b>CpuWdtTimeBase:</b> CPU watchdog timer time base. Read-write. Reset: 0. Specifies the time base for the timeout period specified in CpuWdtCountSel.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00b</td><td>1.31ms</td></tr> <tr> <td>01b</td><td>1.28us</td></tr> <tr> <td>10b</td><td>Reserved (5ns)</td></tr> <tr> <td>11b</td><td>Reserved</td></tr> </table>	Value	Description	00b	1.31ms	01b	1.28us	10b	Reserved (5ns)	11b	Reserved
Value	Description										
00b	1.31ms										
01b	1.28us										
10b	Reserved (5ns)										
11b	Reserved										

Figure 17: Register Field Sub-Row for Valid Value Table

The following figure highlights a valid bit table which is used when each bit has a specific function.

55:52	Reserved.										
51:48	<b>SliceMask.</b> Read-write. Reset: 0.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Bit</th><th>Description</th></tr> <tr> <td>[0]</td><td>L3 Slice 0 mask.</td></tr> <tr> <td>[1]</td><td>L3 Slice 1 mask.</td></tr> <tr> <td>[2]</td><td>L3 Slice 2 mask.</td></tr> <tr> <td>[3]</td><td>L3 Slice 3 mask.</td></tr> </table>	Bit	Description	[0]	L3 Slice 0 mask.	[1]	L3 Slice 1 mask.	[2]	L3 Slice 2 mask.	[3]	L3 Slice 3 mask.
Bit	Description										
[0]	L3 Slice 0 mask.										
[1]	L3 Slice 1 mask.										
[2]	L3 Slice 2 mask.										
[3]	L3 Slice 3 mask.										

Figure 18: Register Field Sub-Row for Valid Bit Table

#### 1.4.4.9 Field Name is Reserved

When a register field name is Reserved, and it does not explicitly specify an access type, then the implied access type is "write-as-read".

- Reads must not depend on the read value.
- Writes must only write the value that was read.

#### 1.4.4.10 Field Access Type

The AccessType keyword is optional and specifies the access type for a register field. The access type for a field is a comma separated list of the following access types.

Table 8: AccessType Definitions

Term	Description
<b>Read-only</b>	Readable; writes are ignored.
<b>Read-write</b>	Readable and writable.
<b>Read</b>	Readable; must be associated with one of the following {Write-once, Write-1-only, Write-1-to-clear, Error-on-write}.
<b>Write-once</b>	Capable of being written once; all subsequent writes have no effect. If not associated with Read, then reads are undefined.
<b>Write-only</b>	Writable. Reads are undefined.
<b>Write-1-only</b>	Writing a 1 sets to a 1; Writing a 0 has no effect. If not associated with Read, then reads are undefined.
<b>Write-1-to-clear</b>	Writing a 1 clears to a 0; Writing a 0 has no effect. If not associated with Read, then reads are undefined.
<b>Write-0-only</b>	Writing a 0 clears to a 0; Writing a 1 has no effect. If not associated with Read, then reads are undefined.
<b>Error-on-read</b>	Error occurs on read.
<b>Error-on-write</b>	Error occurs on write.
<b>Error-on-write-0</b>	Error occurs on bitwise write of 0.
<b>Error-on-write-1</b>	Error occurs on bitwise write of 1.
<b>Inaccessible</b>	Not readable or writable (e.g., Hide ? Inaccessible : Read-Write).
<b>Configurable</b>	Indicates that the access type is configurable as described by the documentation.
<b>Unpredictable</b>	The behavior of both reads and writes is unpredictable.
<b>Reserved-write-as-1</b>	Reads are undefined. Must always write 1.
<b>Reserved-write-as-0</b>	Reads are undefined. Must always write 0.
<b>Volatile</b>	Indicates that a register field value may be modified by hardware, firmware, or microcode when fetching the first instruction and/or might have read or write side effects. No read may depend on the results of a previous read and no write may be omitted based on the value of a previous read or write. Not volatile indicates that software may service a read from the results of a previous read and that a write may be dropped if it's value matches the value previously read or written.

#### 1.4.4.10.1 Conditional Access Type Expression

The ternary operator can be used to express an access type that is conditional on an expression that can contain any of the following:

- A register field value
- A constant
- A definition

#### 1.4.4.11 Field Reset

The Reset keyword is optional and specifies the value for a register field at the time that hardware exits reset, before firmware initialization initiates.

Unless preceded by one of the following prefixes, the reset value is called warm reset and the value is applied at both warm and cold reset.

*Table 9: Reset Type Definitions*

Type	Description
Cold	Cold reset. The value is applied only at cold reset.
Fixed	The read value that applies at all times.

#### 1.4.4.12 Field Initialization

The Init keyword is optional and specifies an initialization recommendation for a register field.

If present, then there is an optional prefix that specifies the owner of the initialization. See Table 10 [Init Type Definitions].

- Example: Init: BIOS,2'b00. //A initialization recommendation for a field to be programmed by BIOS.

*Table 10: Init Type Definitions*

Type	Description
BIOS	Initialized by AMD provided AMD Generic Encapsulated Software Architecture (AGESA™) x86 software.
SBIOS	Initialized by OEM or IBV provided x86 software, also called Platform BIOS.
OS	Initialized by OS or Driver.

#### 1.4.4.13 Field Check

The Check keyword is optional and specifies the value that is recommended for firmware/software to write for a register field. It is a recommendation, not a requirement, and may not under all circumstances be what software programs.

#### 1.4.4.14 Field Valid Values

A register can optionally have either a valid values table or a valid bit table:

- A valid values table specifies the definition for specific field values.
- A valid bit table specifies the definition for specific field bits.

### 1.4.5 Revision History and Change Bar Notation

If a set of PDFs is generated to show differences with respect to a previous release, then:

- A revision history table is generated and exists before the Overview section. (highlighted in red in the following figure)
- The top line indicates what release the changes are in reference to.
- Changes in the revision history have 3 types:
  - Add. A heading, register or field is added.
  - Delete. A heading, register, or field is deleted.
  - Updated. A heading, register, or field is updated.

## Revision History

PPR Revision 3.05 Changes with respect to 3.02, Apr 9, 2024, PUB release:

- 1.2 [Reference Documents]: Updated.
- 1.4.3.2 [Physical Mnemonic]: Updated.
- 1.4.4.5.4 [BAR Access Method]: Updated.
- 1.4.4.5.4.1 [BAR as a Register Reference]: Updated.
- 1.4.4.5.6 [Data Port Access Method]: Updated.
- 1.8.1 [Features]: Updated.
- 1.8.2 [PCI Device ID Assignments]: Added.
- 2.1.1.1 [Core Definitions]: Updated.
- Core::X86::Apic::InterruptRequest: Updated.
- Core::X86::Apic::TriggerMode: Updated.
- Core::X86::Cpuid::CachePropEc3: Updated.
- Core::X86::Cpuid::ProcExtStateEnumEax00: Updated.
- Core::X86::Msr::CpuWdtCfg: Updated.
- Core::X86::Msr::MmioCfgBaseAddr: Updated.
- Core::X86::Pmc::Core::LsAnyFillsFromSys: Updated.
- Core::X86::Pmc::Core::LsDispatch: Updated.
- Core::X86::Pmc::Core::LsSmrRx: Updated.
- Core::X86::Pmc::Core::LsWebCloseFlush: Added.
- MCA::L5::MCA\_CTL\_MASK\_L5: Updated.
- MCA::IF::MCA\_CTL\_MASK\_IF: Updated.
- MCA::L2::MCA\_CTL\_MASK\_L2: Updated.
- MCA::FP::MCA\_CTL\_MASK\_FP: Updated.
- MCA::L3::MCA\_CTL\_MASK\_L3: Updated.
- MCA::L3::MCA\_IPID\_L3: Updated.
- MCA::CS::MCA\_IPID\_CS: Updated.
- MCA::PIE::MCA\_IPID\_PIE: Updated.
- MCA::UMC::MCA\_IPID\_UMC: Updated.
- 5.4.2.2.1 [SB-RMI Mailbox Sequence]: Updated.
- 5.6 [SB-RMI Registers]: Updated.
- 6.2.4 [Atomic Read Mechanism]: Updated.

### Bookmarks

- Legal Notices
- List of Chapters
- Table of Contents
- List of Figures
- List of Tables
- Revision History**
- 1 Overview
- 2 Core Complex (CCX)
- 3 Reliability, Availability, and Serviceability (RAS) Features
- 4 System Management Unit (SMU)
- 5 Advanced Platform Management Link (APML)
- 6 SB Temperature Sensor Interface (SB-TSI)
- List of Namespaces
- List of Definitions

Figure 19: Revision History Format Example

The following diagram shows the notation for changes:

- If a change exists on a line then a thin vertical bar is rendered in the left margin, as circled in blue.
- Deleted text is indicated with amber and strike-through, as circled in red.
- Added text is indicated with amber and underlined, as circled in green.

### 5.6 SB-RMI Registers

 Reads to unimplemented registers may return ~~00h~~ on zero value. Writes to unimplemented registers are discarded.

Figure 20: Change Notation Example

## 1.5 Definitions

*Table 11: Definitions*

Term	Description
<b>AGESA™</b>	AMD Generic Encapsulated Software Architecture.
<b>AP</b>	Applications Processor.
<b>APML</b>	Advanced Platform Management Link.
<b>BCD</b>	Binary Coded Decimal number format.
<b>BCS</b>	Base Configuration Space.
<b>BIST</b>	Built-In Self-Test. Hardware within the processor that generates test patterns and verifies that they are stored correctly (in the case of memories) or received without error (in the case of links).
<b>Boot VID</b>	Boot Voltage ID. This is the VDD and VDDNB voltage level that the processor requests from the external voltage regulator during the initial phase of the cold boot sequence.
<b>C-states</b>	These are ACPI defined core power states. C0 is operational. All other C-states are low-power states in which the processor is not executing code. See docACPI.
<b>Cold reset</b>	PWROK is de-asserted and RESET_L is asserted.
<b>COF</b>	Current operating frequency of a given clock domain.
<b>DID</b>	Divisor Identifier. Specifies the post-PLL divisor used to reduce the COF.
<b>Doubleword</b>	A 32-bit value.
<b>DW</b>	Doubleword.
<b>ECS</b>	Extended Configuration Space.
<b>FCH</b>	The integrated platform subsystem that contains the IO interfaces and bridges them to the system BIOS. Previously included in the Southbridge.
<b>FID</b>	Frequency Identifier. Specifies the PLL frequency multiplier for a given clock domain.
<b>GB</b>	Gbyte or Gigabyte; 1,073,741,824 bytes.
<b>GT/s</b>	Giga-Transfers per second.
<b>IFCM</b>	Isochronous flow-control mode, as defined in the link specification.
<b>IO configuration</b>	Access to configuration space through IO ports CF8h and CFCh.
<b>IP</b>	In electronic design, a semiconductor Intellectual Property, IP, or IP block is a reusable unit of logic, cell, or integrated circuit layout design that is the intellectual property of one party.
<b>KB</b>	Kbyte or Kilobyte; 1024 bytes.
<b>Master abort</b>	This is a PCI-defined term that is applied to transactions on other than PCI buses. It indicates that the transaction is terminated without affecting the intended target; Reads return all 1s; Writes are discarded; the master abort error code is returned in the response, if applicable; master abort error bits are set if applicable.
<b>MB</b>	Megabyte; 1024 KB.
<b>MMIO</b>	Memory-Mapped Input-Output range. This is physical address space that is mapped to the IO functions such as the IO links or MMIO configuration.
<b>MMIO configuration</b>	Access to configuration space through memory space.
<b>OW</b>	Octword. An 128-bit value.
<b>PCIe®</b>	PCI Express.
<b>PCS</b>	Physical Coding Sublayer.
<b>Processor</b>	System on Chip (SoC) covered by this PPR. See 1.8 [Processor Overview].
<b>P-state</b>	Performance state.
<b>QW</b>	Quadword. A 64-bit value.
<b>RAS</b>	Reliability, availability and serviceability (industry term). See 3.1 [Machine Check Architecture].
<b>REFCLK</b>	Reference clock. Refers to the clock frequency (100 MHz) or the clock period (10 ns) depending on the context used.
<b>RX</b>	Receiver.
<b>Shutdown</b>	A state in which the affected core waits for either INIT, RESET, or NMI. When shutdown state is entered, a shutdown special cycle is sent on the IO links.

<b>SMAF</b>	System Management Action Field. This is the code passed from the SMC to the processors in STPCLK assertion messages.
<b>SMC</b>	System Management Controller. This is the platform device that communicates system management state information to the processor through an IO link, typically the system IO hub.
<b>Speculative event</b>	A performance monitor event counter that counts all occurrences of the event even if the event occurs during speculative code execution.
<b>SSC</b>	Spread Spectrum Clocking.
<b>TDC</b>	Thermal Design Current.
<b>TDP</b>	Thermal Design Power. A power consumption parameter that is used in conjunction with thermal specifications to design appropriate cooling solutions for the processor.
<b>TOM</b>	Top of Memory.
<b>TOM2</b>	Top of extended Memory.
<b>TX</b>	Transmitter.
<b>UMI</b>	Unified Media Interface. The link between the processor and the FCH.
<b>VID</b>	Voltage level identifier.
<b>Warm reset</b>	RESET_L is asserted only (while PWROK stays high).
<b>XBAR</b>	Cross bar; command packet switch.

## 1.6 Changes Between Revisions and Product Variations

### 1.6.1 Revision Conventions

The processor revision is specified by CPUID\_Fn00000001\_EAX (FamModStep) or CPUID\_Fn80000001\_EAX (FamModStepExt). This document uses a revision letter instead of specific model numbers. Where applicable, the processor stepping is indicated after the revision letter. All behavior marked with a revision letter apply to future revisions unless they are superseded by a change in a later revision. See the revision guide in 1.2 [Reference Documents] for additional information about revision determination.

## 1.7 Package

### 1.7.1 Package type

The following packages are supported.

*Table 12: Package Definitions*

<b>Term</b>	<b>Description</b>
<b>AM5</b>	Desktop, single socket. For client desktop platform (LGA) with DDR5. AM5 = (Core::X86::Cpuid::BrandId[PkgType] == 00h).
<b>FL1</b>	Notebook and mobile workstations. FL1 = (Core::X86::Cpuid::BrandId[PkgType] == 01h).

## 1.8 Processor Overview

### 1.8.1 Features

Family 1Ah Models 40h-4Fh are microprocessor System-On-a-Chip (SOC) multi-chip module (MCM). It is built using a



mixed-technology chiplet approach - up to two core/cache complex dies (CCD) and a single I/O die (IOD).

2 Core Complex (CCX)

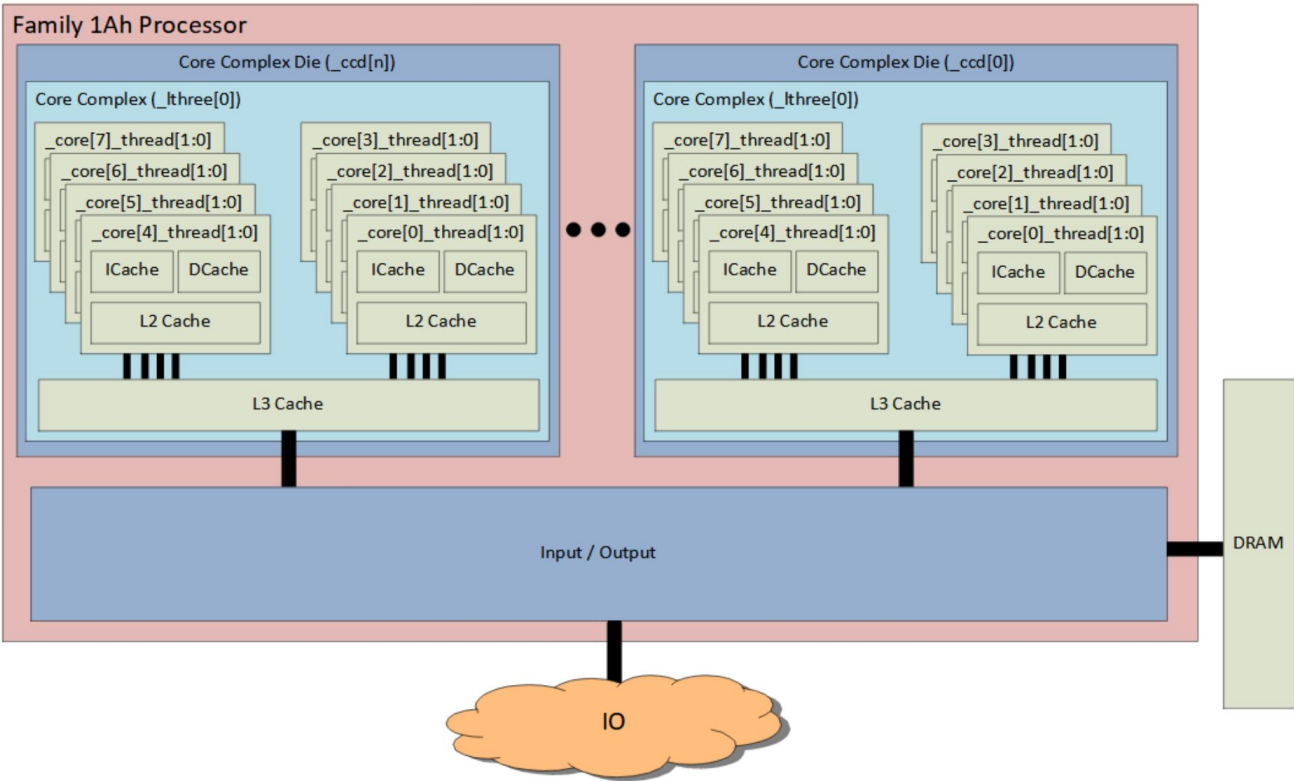


Figure 21: Overview of Family 1A Core Component Numbering

## **2.1 Processor x86 Core**

### **2.1.1 Core Functional Information**

#### **2.1.1.1 Core Definitions**

*Table 13: Definitions*

Term	Description
<b>BSC</b>	Boot strap core. Core 0 of the BSP.
<b>BSP</b>	Boot strap processor.
<b>Canonical-address</b>	An address in which the state of the most-significant implemented bit is duplicated in all the remaining higher-order bits, up to bit[63].
<b>CCX</b>	Core Complex where more than one core shares L3 resources.
<b>CMP</b>	Specifies the core number.
<b>Core</b>	The instruction execution unit of the processor when the term Core is used in a x86 core context.
<b>CoreCOF</b>	Core current operating frequency in MHz. CoreCOF = Core::X86::Msr::PStateDef[CpuFid[11:0]] * 5MHz.
<b>CPL</b>	Current Privilege Level of the running task when the term CPL is used in a x86 core context.
<b>CpuCoreNum</b>	Specifies the core number.
<b>#GP</b>	A general-protection exception.
<b>#GP(0)</b>	Notation indicating a general-protection exception (#GP) with error code of 0.
<b>HWPF</b>	Hardware Prefetcher.
<b>IBS</b>	Instruction based sampling.
<b>IO configuration</b>	Access to configuration space through IO ports CF8h and CFCh.
<b>IORR</b>	IO range register.
<b>L1 cache</b>	The level 1 caches (instruction cache and the data cache).
<b>L2 cache</b>	The level 2 caches.
<b>L3</b>	Level 3 Cache. The L3 term is also in Addrmaps to enumerate CCX units.
<b>L3 cache</b>	Level 3 Cache.
<b>Linear (virtual) address</b>	The address generated by a core after the segment is applied.
<b>LINT</b>	Local interrupt.
<b>Logical address</b>	The address generated by a core before the segment is applied.
<b>LRU</b>	Least recently used.
<b>LVT</b>	Local vector table. A collection of APIC registers that define interrupts for local events (e.g., APIC[530:500] [Extended Interrupt [3:0] Local Vector Table]).
<b>Macro-op</b>	The front-end of the pipeline breaks instructions into macro-ops and transfers (dispatches) them to the back-end of the pipeline for scheduling and execution. See Software Optimization Guide.
<b>Micro-op</b>	Processor schedulers break down macro-ops into sequences of even simpler instructions called micro-ops, each of which specifies a single primitive operation. See Software Optimization Guide.
<b>NBC</b>	NBC=(CPUID Fn00000001_EBX[LocalApicId[3:0]] == 0). Node Base Core. The lowest numbered core in the node.
<b>MTRR</b>	Memory-type range register. The MTRRs specify the type of memory associated with various memory ranges.
<b>MPB</b>	Microcode patch block.
<b>NTA</b>	Non-Temporal Access.
<b>PPIN</b>	Protected Processor Inventory Number.
<b>PTE</b>	Page table entry.
<b>SMI</b>	System management interrupt.
<b>SMM</b>	System Management Mode.
<b>SMT</b>	Simultaneous multithreading. See Core::X86::CpuId::CoreId[ThreadsPerCore].
<b>Speculative event</b>	A performance monitor event counter that counts all occurrences of the event even if the event occurs during speculative code execution.
<b>SVM</b>	Secure virtual machine.
<b>Thread</b>	One architectural context for instruction execution.

<b>VMPL</b>	Virtual Machine Privilege Level.
<b>WDT</b>	Watchdog timer. A timer that detects activity and triggers an error if a specified period of time expires without the activity.
<b>X2APICEN</b>	x2 APIC is enabled. X2APICEN = (Core::X86::Msrr::APIC_BAR[ApicEn] && Core::X86::Msrr::APIC_BAR[x2ApicEn]).

## 2.1.2 Secure Virtual Machine Mode (SVM)

Support for SVM mode is indicated by Core::X86::Cpuid::FeatureExtIdEcX[SVM].

### 2.1.2.1 BIOS support for SVM Disable

The BIOS should include the following user setup options to enable and disable AMD Virtualization™ technology.

#### 2.1.2.1.1 Enable AMD Virtualization™

- Core::X86::Msrr::VM\_CR[SvmeDisable] = 0.
- Core::X86::Msrr::VM\_CR[Lock] = 1.
- Core::X86::Msrr::SvmLockKey[SvmLockKey] = 0000\_0000\_0000\_0000h.

#### 2.1.2.1.2 Disable AMD Virtualization™

- Core::X86::Msrr::SvmLockKey[SvmLockKey] = 0000\_0000\_0000\_0000h.
- Core::X86::Msrr::VM\_CR[SvmeDisable] = 1.
- Core::X86::Msrr::VM\_CR[Lock] = 1.

The BIOS may also include the following user setup options to disable AMD Virtualization technology.

#### 2.1.2.1.3 Disable AMD Virtualization™, with a user supplied key

- Core::X86::Msrr::VM\_CR[SvmeDisable] = 1.
- Core::X86::Msrr::VM\_CR[Lock] = 1.
- Core::X86::Msrr::SvmLockKey[SvmLockKey] programmed with value supplied by user. This value should be stored in NVRAM.

## 2.1.3 Microcode Patching

### 2.1.3.1 Microcode Patching Overview

The processor contains a small amount of RAM for implementing microcode patches. This patch RAM is loaded by a microcode routine (the Patch RAM Loader) that is part of the normal microcode contained in ROM. The patch RAM contains 64 microlines. Microlines are sets of processor microcode operations which are grouped to form a line of microcode. When the processor powers up, it uses its internal ROM microcode. If no patches are installed, the processor only executes microcode from the ROM.

The Patch RAM Loader function is called via a write to Core::X86::Msrr::PATCH\_LOADER. The patch loader downloads microcode from the Microcode Patch Block (MPB) stored in memory into the processor patch RAM.

### 2.1.3.2 Microcode Patch Block (MPB)

The MPB is 14368 bytes in length and consists of two parts. The first part is a 32-byte header. The second part contains code, including the encrypted patch data. The properties of the patch allow for multiple consistency checks after decryption. If any consistency check fails, a #GP is generated. The patch data is encrypted to prevent unauthorized use of the patch mechanism.

Table 14: MPB Definition

Field	Byte Offset	Byte Length	Description
MPB_DATE	0000h	4	A doubleword for "mmddyyyy" in BCD format
Patch Level	0004h	4	A unique patch level as defined by the Revision Guide
MPB_LOADER_ID	0008h	2	A unique ID used for checking if the update is successful after the microcode patch function is executed.
MPB_SIZE	000Ah	2	
Reserved	000Ch	4	
MPB_NB_ID	0010h	4	The BIOS must match the chipset 1 device ID with this field before executing the microcode patch. A "0" means the patch is not chipset specific.
MPB_SB_ID	0014h	4	The BIOS must match the chipset 2 device ID with this field before executing the microcode patch. A "0" means the patch is not chipset specific.
MPB_REVISION	0018h	4	Microcode Patch Equivalent Processor Id.
MPB_BIOS_REVISION	001Ch	1	
Load Control	001Dh	1	Bit[0]: SerializedLoad: =0 (default), =1 The patch is required to be loaded serially across all cores in the system. Bit[1]: LoadOnBothThreads: = 0 (default) The patch is required to be loaded on both threads on each core in the system. =1 The patch may be loaded on one thread on each core in the system. Bits[7:2]: Reserved.
NEXT_MPB_OFFSET	001Eh	2	For MPB chaining. Offset of the next MPB in multiples of 16byte. An offset =0 indicates that no other MPB follows the current MPB.
Patch Data	0020h	14336	Patch Data.
Total MPB Size		14368	

### 2.1.4 Effective Frequency

The effective frequency interface allows software to discern the average, or effective, frequency of a given core over a configurable window of time. This provides software a measure of actual performance rather than forcing software to assume the current frequency of the core is the frequency of the last P-state requested. Core::X86::Msr::MPERF is incremented by hardware at the P0 frequency while the core is in C0. Core::X86::Msr::APERF increments in proportion to the actual number of core clocks cycles while the core is in C0.

The following procedure calculates effective frequency using Core::X86::Msr::MPERF and Core::X86::Msr::APERF:

1. At some point in time, write 0 to both MSRs.
2. At some later point in time, read both MSRs.
3. Effective frequency = (value read from Core::X86::Msr::APERF / value read from Core::X86::Msr::MPERF) \* P0 frequency.

## Additional notes:

- The amount of time that elapses between steps 1 and 2 is determined by software.
- It is software's responsibility to disable interrupts or any other events that may occur in between the Write of Core::X86::Msrb::MPERF and the Write of Core::X86::Msrb::APERF in step 1 or between the Read of Core::X86::Msrb::MPERF and the Read of Core::X86::Msrb::APERF in step 2.
- The behavior of Core::X86::Msrb::MPERF and Core::X86::Msrb::APERF may be modified by Core::X86::Msrb::HWCR[EffFreqCntMwait].
- The effective frequency interface provides +/- 50MHz accuracy if the following constraints are met:
  - Effective frequency is read at most one time per millisecond.
  - When reading or writing Core::X86::Msrb::MPERF and Core::X86::Msrb::APERF software executes only MOV instructions, and no more than 3 MOV instructions, between the two RDMSR or WRMSR instructions.
  - Core::X86::Msrb::MPERF and Core::X86::Msrb::APERF are invalid if an overflow occurs.

## 2.1.5 Address Space

### 2.1.5.1 Virtual Address Space

The processor supports 48-bit address bits of virtual memory space (256 TB) as indicated by Core::X86::Cpuid::LongModeInfo.

### 2.1.5.2 Physical Address Space

The processor supports a 48-bit physical address space. See Core::X86::Cpuid::LongModeInfo. The processor master aborts the following upper-address transactions (to address PhysAddr):

- Link or core requests with non-zero PhysAddr[63:48].

### 2.1.5.3 System Address Map

The processor defines a Reserved memory address region starting at FFFD\_0000\_0000h and extending up to FFFF\_FFFF\_FFFFh. System software must not map memory into this region. Downstream host accesses to the Reserved address region results in a page fault. Upstream system device accesses to the reserved address region results in an undefined operation.

#### 2.1.5.3.1 Memory Access to the Physical Address Space

All memory accesses to the physical address space from a core are sent to its associated Data Fabric (DF). All memory accesses from a link are routed through the DF. An IO link access to physical address space indicates to the DF the cache attribute (Coherent or Non-coherent, based on bit[0] of the Sized Read and Write commands).

A core access to physical address space has two important attributes that must be determined before issuing the access to the Northbridge: the memory type (e.g., WB, WC, UC; as described in the MTRRs) and the access destination (DRAM or MMIO).

If the memory map maps a region as DRAM that is not populated with real storage behind it, then that area of DRAM must be mapped as UC memtype.

This mechanism is managed by the BIOS and does not require any setup or changes by system software.

### 2.1.5.3.1.1 Determining Memory Type

The memory type for a core access is determined by the highest priority of the following ranges that the access falls in: 1=Lowest priority.

1. The memory type as determined by architectural mechanisms.
  - See the docAPM2 chapter titled "Memory System", sections "Memory-Type Range Registers" and "Page-Attribute Table Mechanism".
  - See the docAPM2 chapter titled "Nested Paging", section "Combining Memory Types, MTRRs".
  - See Core::X86::Msrr::MTRRdefType, Core::X86::Msrr::MtrrVarBase, Core::X86::Msrr::MtrrVarMask, Core::X86::Msrr::MtrrFix\_64K and Core::X86::Msrr::MtrrFix\_16K\_0 through Core::X86::Msrr::MtrrFix\_4K\_7.
2. TSeg & ASeg SMM mechanism (See Core::X86::Msrr::SMMAddr and Core::X86::Msrr::SMMMask).
3. CR0[CD]: If (CR0[CD] == 1) then MemType = CD.
4. MMIO configuration space, APIC space.
  - MMIO APIC space and MMIO config space must not overlap.
  - MemType = UC.
5. If ("In SMM Mode"&& ~((Core::X86::Msrr::SMMMask[AValid] && "The address falls within the ASeg region") || (Core::X86::Msrr::SMMMask[TValid] && "The address falls within the TSeg region")) then MemType = CD.

### 2.1.6 Configuration Space

PCI-defined configuration space was originally defined to allow up to 256 bytes of register space for each function of each device; these first 256 bytes are called base configuration space (BCS). It was expanded to support up to 4096 bytes per function; bytes 256 through 4095 are called extended configuration space (ECS).

The processor includes configuration space registers located in both BCS and ECS. Processor configuration space is accessed through bus 0, devices 18h to 1Fh, where device 18h corresponds to node 0 and device 1Fh corresponds to node 7. See 2.1.6.3 [Processor Configuration Space].

Configuration space is accessed by the processor through two methods as follows:

- IO-space configuration: IO instructions to addresses CF8h and CFCh.
  - Enabled through IO::IoCfgAddr[ConfigEn], which allows access to BCS.
  - Use of IO-space configuration can be programmed to generate GP faults through Core::X86::Msrr::HWCR[IoCfgGpFault].
  - SMI trapping for these accesses is specified by Core::X86::Msrr::SMI\_ON\_IO\_TRAP\_CTL\_STS and Core::X86::Msrr::SMI\_ON\_IO\_TRAP.
- Memory Mapped IO (MMIO) configuration: configuration space is a region of memory space.
  - The base address and size of this range is specified by Core::X86::Msrr::MmioCfgBaseAddr. The size is controlled by the number of configuration-space bus numbers supported by the system. Accesses to this range are converted configuration space as follows:
- Address[31:0] = {0h, segment[6:0], bus[7:0], device[4:0], function[2:0], offset[11:0]}.

The BIOS may use either configuration space access mechanism during boot. Before booting the OS, BIOS must disable IO access to ECS, enable MMIO configuration and build an ACPI defined MCFG table. BIOS ACPI code must use MMIO to access configuration space.

#### 2.1.6.1 Memory Mapped IO (MMIO) Configuration Coding Requirements

MMIO configuration space accesses must use the uncachable (UC) memory type.

Instructions used to read MMIO configuration space are required to take the following form:

```
mov eax/ax/al, any_address_mode;
```



Instructions used to write MMIO configuration space are required to take the following form:

```
mov any_address_mode, eax/ax/al;
```

No other source/target registers may be used other than eax/ax/al.

In addition, all such accesses are required not to cross any naturally aligned DW boundary. Access to MMIO configuration space registers that do not meet these requirements result in undefined behavior.

### **2.1.6.2 MMIO Configuration Ordering**

Since MMIO configuration cycles are not serializing in the way that IO configuration cycles are, their ordering rules relative to posted may result in unexpected behavior.

Therefore, processor MMIO configuration space is designed to match the following ordering relationship that exists naturally with IO-space configuration: if a core generates a configuration cycle followed by a posted Write cycle, then the posted Write is held in the processor until the configuration cycle completes. As a result, any unexpected behavior that might have resulted if the posted Write cycle were to pass MMIO configuration cycle is avoided.

### **2.1.6.3 Processor Configuration Space**

Accesses to unimplemented registers of implemented functions are ignored: Writes dropped; Reads return 0. Accesses to unimplemented functions also ignored: Writes are dropped; however, Reads return all F's. The processor does not log any master abort events for accesses to unimplemented registers or functions.

Accesses to device numbers of devices not implemented in the processor are routed based on the configuration map registers. If such requests are master aborted, then the processor can log the event.

### **2.1.7 PCI Configuration Legacy Access**

**IOx0CF8 [IO-Space Configuration Address] (IO::IoCfgAddr)**

Read-write. Reset: 0000\_0000h.

IO::IoCfgAddr, and IO::IoCfgData are used to access system configuration space, as defined by the PCI specification. IO::IoCfgAddr provides the address register and IO::IoCfgData provides the data port. Software sets up the configuration address by writing to IO::IoCfgAddr. Then, when an access is made to IO::IoCfgData, the processor generates the corresponding configuration access to the address specified in IO::IoCfgAddr. See 2.1.6 [Configuration Space].

IO::IoCfgAddr may only be accessed through aligned, DW IO Reads and Writes; otherwise, the accesses are passed to the appropriate IO link. Accesses to IO::IoCfgAddr and IO::IoCfgData received from an IO link are treated as all other IO transactions received from an IO link. IO::IoCfgAddr and IO::IoCfgData in the processor are not accessible from an IO link.

\_aliasIO; IOx0CF8; IO=0000\_0000h

Bits	Description
31	<b>ConfigEn: configuration space enable.</b> Read-write. Reset: 0. 0=IO Read and Write accesses are passed to the appropriate IO link and no configuration access is generated. 1=IO Read and Write accesses to IO::IoCfgData are translated into configuration cycles at the configuration address specified by this register.
30:28	Reserved.
27:24	<b>ExtRegNo: extended register number.</b> Read-write. Reset: 0h. ExtRegNo provides bits[11:8] and RegNo provides bits[7:2] of the byte address of the configuration register.
23:16	<b>BusNo: bus number.</b> Read-write. Reset: 00h. Specifies the bus number of the configuration cycle.
15:11	<b>Device: device number.</b> Read-write. Reset: 00h. Specifies the device number of the configuration cycle.
10:8	<b>Function.</b> Read-write. Reset: 0h. Specifies the function number of the configuration cycle.
7:2	<b>RegNo: register address.</b> Read-write. Reset: 00h. See IO::IoCfgAddr[ExtRegNo].
1:0	Reserved.

**IOx0CFC [IO-Space Configuration Data Port] (IO::IoCfgData)**

Read-write. Reset: 0000\_0000h.

\_aliasIO; IOx0CFC; IO=0000\_0000h

Bits	Description
31:0	<b>Data.</b> Read-write. Reset: 0000_0000h. See IO::IoCfgAddr.

**2.1.8 System Software Interaction With SMT Enabled**

If Core::X86::Cpuid::CoreId[ThreadsPerCore] > 0, then SMT is enabled in all cores in the system. When SMT is enabled, the resources of each core are dynamically balanced among the hardware threads executing on that core. The number of hardware threads (hereafter "threads") supported by a single core when SMT is enabled is reported in Core::X86::Cpuid::CoreId[ThreadsPerCore]. System software that is SMT-aware may take advantage of the knowledge that core resources are being shared among multiple threads when scheduling tasks to be run by each thread on each core. System software that is not SMT-aware sees each thread as an independent core.

## 2.1.9 Register Sharing

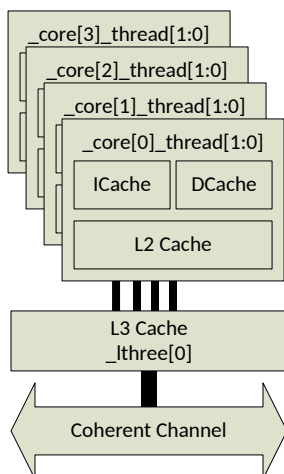


Figure 22: Register Sharing Domains

### MSR0000\_0010 [Time Stamp Counter] (TSC)

Read-write, Volatile. Reset: 0000_0000_0000_0000h.	
Core::X86::Msr::TSC_lthree0_core[3:0]_thread[1:0]; MSR00000010	
Bits	Description
63:0	<b>TSC: time stamp counter.</b> Read-write, Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the TSC P0 frequency based value (as if TSCRatio == 1.0) when (TSCRatio != 1.0).

Figure 23: Instance Parameters

Instances of core registers are designated as `ccd[n:0]_lthree[n:0]_core[n:0]_thread[1:0]`. Core registers may be shared at various levels of hierarchy as one register instance per node, per L3 complex, per core or per thread. The absence of the instance parameter `_thread[1:0]` signifies that there is not a specific instance of said register per thread and thus the register is shared between thread[1] and thread[0]. Similarly, the absence of the instance parameter `_core[n:0]` signifies that there is not a specific instance of said register per core and thus the register is shared by all cores in that L3 complex, and so on. The absence of instance parameters indicate there is one shared register at the node level. Software must coordinate writing to shared registers with other threads in the same sharing hierarchy level.

## 2.1.10 Timers

Each core includes the following timers. These timers do not vary in frequency regardless of the current P-state or C-state.

- Core::X86::Msr::TSC; the TSC increments at the rate specified by the P0 Pstate.
- The APIC timer (Core::X86::Apic::TimerInitialCount and Core::X86::Apic::TimerCurrentCount), which increments at the rate of  $2 \times \text{CLKIN}$ ; the APIC timer may increment in units of between 1 and 8.

## 2.1.11 Interrupts

### 2.1.11.1 System Management Mode (SMM)

System management mode (SMM) is typically used for system control activities such as power management. These activities are typically transparent to the operating system.

#### 2.1.11.1.1 SMM Overview

SMM is entered by a core on the next instruction boundary after a system management interrupt (SMI) is received and recognized. A core may be programmed to broadcast a special cycle to the system, indicating that it is entering SMM mode. The core then saves its state into the SMM memory state save area and jumps to the SMI service routine (or SMI handler). The pointer to the SMI handler is specified by MSRs. The code and data for the SMI handler are stored in the SMM memory area, which may be isolated from the main memory accesses.

The core returns from SMM by executing the RSM instruction from the SMI handler. The core restores its state from the SMM state save area and resumes execution of the instruction following the point where it entered SMM. The core may be programmed to broadcast a special bus cycle to the system, indicating that it is exiting SMM mode.

#### 2.1.11.1.2 Mode and Default Register Values

The software environment after entering SMM has the following characteristics:

- Addressing and operation is in Real mode.
  - A far jump, call or return in the SMI handler can only address the lower 1M of memory, unless the SMI handler first switches to protected mode.
  - If (Core::X86::Msrb::SMM\_BASE[SmmBase] >= 0010\_0000h) then:
    - The value of the CS selector is undefined upon SMM entry.
    - The undefined CS selector value should not be used as the target of a far jump, call, or return.
- 4-Gbyte segment limits.
- Default 16-bit operand, address, and stack sizes (instruction prefixes can override these defaults).
- Control transfers that do not override the default operand size truncate the EIP to 16 bits.
- Far jumps or calls cannot transfer control to a segment with a base address requiring more than 20 bits, as in Real mode segment-base addressing, unless a change is made into protected mode.
- Interrupt vectors use the Real mode interrupt vector table.
- The IF flag in EFLAGS is cleared (INTR is not recognized).
- The TF flag in EFLAGS is cleared.
- The NMI and INIT interrupts are masked.
- Debug register DR7 is cleared (debug traps are disabled).

The SMM base address is specified by Core::X86::Msrb::SMM\_BASE[SmmBase]. Important offsets to the base address pointer are:

- Core::X86::Msrb::SMM\_BASE[SmmBase] + 8000h: SMI handler entry point.
- Core::X86::Msrb::SMM\_BASE[SmmBase] + FE00h - FFFFh: SMM state save area.

#### 2.1.11.1.3 SMI Sources And Delivery

The processor accepts SMIs as link-defined interrupt messages only. The core/node destination of these SMIs is a function of the destination field of these messages. However, the expectation is that all such SMI messages are specified to be delivered globally (to all cores of all nodes).

There are also several local events that can trigger SMIs. However, these local events do not generate SMIs directly. Each of them triggers a programmable IO cycle that is expected to target the SMI command port in the IO hub and trigger a global SMI interrupt message back to the coherent fabric.

Local sources of SMI events that generate the IO cycle specified in Core::X86::Msr::SmiTrigIoCycle are:

- In the core, as specified by:
  - Core::X86::Msr::McExcepRedir.
  - Core::X86::Msr::SMI\_ON\_IO\_TRAP.
- All local APIC LVT registers programmed to generate SMIs.

The status for these are stored in Core::X86::Smm::LocalSmiStatus.

#### 2.1.11.1.4 SMM Initial State

After storing the save state, execution starts at Core::X86::Msr::SMM\_BASE[SmmBase] + 08000h. The SMM initial state is specified in the following table.

Table 15: SMM Initial State

Register	SMM Initial State
CS	SmmBase[19:4]
DS	0000h
ES	0000h
FS	0000h
GS	0000h
SS	0000h
General-Purpose Registers	Unmodified.
EFLAGS	0000_0002h
RIP	0000_0000_0000_8000h
CR0	Bits[0,2,3,31] cleared (PE, EM, TS, and PG); remainder is unmodified.
CR4	0000_0000_0000_0000h
GDTR	Unmodified.
LDTR	Unmodified.
IDTR	Unmodified.
TR	Unmodified.
DR6	Unmodified.
DR7	0000_0000_0000_0400h
EFER	All bits are cleared except bit[12] (SVME) which is unmodified.

#### 2.1.11.1.5 SMM Save State

In the following table, the offset field provides the offset from the SMM base address specified by Core::X86::Msr::SMM\_BASE[SmmBase].

Table 16: SMM Save State

Offset	Size	Contents	Access
FE00h	Word	ES	Read-only
FE02h	6 Bytes	Reserved	

FE08h	Quadword		Descriptor in memory format	
FE10h	Word	CS	Selector	Read-only
FE12h	6 Bytes		Reserved	
FE18h	Quadword		Descriptor in memory format	
FE20h	Word	SS	Selector	Read-only
FE22h	6 Bytes		Reserved	
FE28h	Quadword		Descriptor in memory format	
FE30h	Word	DS	Selector	Read-only
FE32h	6 Bytes		Reserved	
FE38h	Quadword		Descriptor in memory form	
FE40h	Word	FS	Selector	Read-only
FE42h	2 Bytes		Reserved	
FE44h	Doublewor d		FS Base {16'b[47], 47:32}(note 1)	
FE48h	Quadword		Descriptor in memory format	
FE50h	Word	GS	Selector	Read-only
FE52h	2 Bytes		Reserved	
FE54h	Doublewor d		GS Base {16'b[47], 47:32}(note 1)	
FE58h	Quadword		Descriptor in memory format	
FE60h	4 Bytes	GDTR	Reserved	Read-only
FE64h	Word		Limit	
FE66h	2 Bytes		Reserved	
FE68h	Quadword		Descriptor in memory format	
FE70h	Word	LDTR	Selector	Read-only
FE72h	Word		Attributes	
FE74h	Doublewor d		Limit	
FE78h	Quadword		Base	
FE80h	4 Bytes	IDTR	Reserved	Read-only
FE84h	Word		Limit	
FE86h	2 Bytes		Reserved	
FE88h	Quadword		Base	
FE90h	Word	TR	Selector	Read-only
FE92h	Word		Attributes	
FE94h	Doublewor d		Limit	
FE98h	Quadword		Base	
FEA0h	Quadword	IO_RESTART_RIP		
FEA8h	Quadword	IO_RESTART_RCX		
FEB0h	Quadword	IO_RESTART_RSI		
FEB8h	Quadword	IO_RESTART_RDI		
FEC0h	Doublewor d	Core::X86::Smm::TrapOffset [SMM IO Trap Offset]		Read-only
FEC4	Doublewor d	Core::X86::Smm::LocalSmiStatus		Read-only
FEC8h	Byte	Core::X86::Smm::IoRestart		Read-write
FEC9h	Byte	Core::X86::Smm::AutoHalt		Read-write
FECAh	Byte	Core::X86::Smm::NmiMask		Read-write

FECBh	5 Bytes	Reserved	
FED0h	Quadword	EFER	Read-only
FED8h	Quadword	Core::X86::Smm::SvmState	Read-only
FEE0h	Quadword	Guest VMCB physical address	Read-only
FEE8h	Quadword	SVM Virtual Interrupt Control	Read-only
FEF0h	16 Bytes	Reserved	
FEFCh	Doubleword	Core::X86::Smm::SmmRevID	Read-only
FF00h	Doubleword	Core::X86::Smm::SmmBase	Read-write
FF04h	28 Bytes	Reserved	
FF20h	Quadword	Guest PAT	Read-only
FF28h	Quadword	Host EFER (note 2)	
FF30h	Quadword	Host CR4 (note 2)	
FF38h	Quadword	Nested CR3 (note 2)	
FF40h	Quadword	Host CR0 (note 2)	
FF48h	Quadword	CR4	
FF50h	Quadword	CR3	
FF58h	Quadword	CR0	
FF60h	Quadword	DR7	Read-only
FF68h	Quadword	DR6	
FF70h	Quadword	RFLAGS	Read-write
FF78h	Quadword	RIP	Read-write
FF80h	Quadword	R15	
FF88h	Quadword	R14	
FF90h	Quadword	R13	
FF98h	Quadword	R12	
FFA0h	Quadword	R11	
FFA8h	Quadword	R10	
FFB0h	Quadword	R9	
FFB8h	Quadword	R8	
FFC0h	Quadword	RDI	Read-write
FFC8h	Quadword	RSI	
FFD0h	Quadword	RBP	
FFD8h	Quadword	RSP	
FFE0h	Quadword	RBX	
FFE8h	Quadword	RDX	
FFF0h	Quadword	RCX	
FFF8h	Quadword	RAX	

## Notes:

1. This notation specifies that bit[47] is replicated in each of the 16 MSBs of the DW (sometimes called sign extended). The 16 LSBs contain bits[47:32].
2. Only used for an SMI in guest mode with nested paging enabled.

The SMI save state includes most of the integer execution unit. Not included in the save state are: the floating-point state, MSRs, and CR2. In order to be used by the SMI handler, these must be saved and restored. The save state is the same, regardless of the operating mode (32-bit or 64-bit).

### 2.1.11.1.6 System Management State

The following are offsets in the SMM save state area.

SMMxFEC0 [SMM IO Trap Offset] (Core::X86::Smm::TrapOffset)	
Read-only, Volatile. Reset: 0000_0000h.	
If the assertion of SMI is recognized on the boundary of an IO instruction, Core::X86::Smm::TrapOffset contains information about that IO instruction. For example, if an IO access targets an unavailable device, the system can assert SMI and trap the IO instruction. Core::X86::Smm::TrapOffset then provides the SMI handler with information about the IO instruction that caused the trap. After the SMI handler takes the appropriate action, it can reconstruct and then re-execute the IO instruction from SMM. Or, more likely, it can use Core::X86::Smm::IoRestart to cause the core to re-execute the IO instruction immediately after resuming from SMM.	
Bits	Description
31:16	<b>Port: trapped IO port address.</b> Read-only, Volatile. Reset: 0000h. This provides the address of the IO instruction.
15:12	<b>BPR: IO breakpoint match.</b> Read-only, Volatile. Reset: 0h.
11	<b>TF: EFLAGS TF value.</b> Read-only, Volatile. Reset: 0.
10:7	Reserved.
6	<b>SZ32: size 32 bits.</b> Read-only, Volatile. Reset: 0. 1=Port access was 32 bits.
5	<b>SZ16: size 16 bits.</b> Read-only, Volatile. Reset: 0. 1=Port access was 16 bits.
4	<b>SZ8: size 8 bits.</b> Read-only, Volatile. Reset: 0. 1=Port access was 8 bits.
3	<b>REP: repeated port access.</b> Read-only, Volatile. Reset: 0.
2	<b>STR: string-based port access.</b> Read-only, Volatile. Reset: 0.
1	<b>V: IO trap word valid.</b> Read-only, Volatile. Reset: 0. 0=The other fields of this offset are not valid. 1=The core entered SMM on an IO instruction boundary; all information in this offset is valid.
0	<b>RW: port access type.</b> Read-only, Volatile. Reset: 0. 0=IO Write (OUT instruction). 1=IO Read (IN instruction).
SMMxFEC4 [Local SMI Status] (Core::X86::Smm::LocalSmiStatus)	
Read-only, Volatile. Reset: 0000_0000h.	
This offset stores status bits associated with SMI sources local to the core. For each of these bits, 1=The associated mechanism generated an SMI.	
Bits	Description
31:9	Reserved.
8	<b>MceRedirSts: machine check exception redirection status.</b> Read-only, Volatile. Reset: 0. This bit is associated with the SMI source specified in Core::X86::Msr::McExcepRedir[RedirSmiEn].
7:4	Reserved.
3:0	<b>IoTrapSts: IO trap status.</b> Read-only, Volatile. Reset: 0h. Each of these bits is associated with each of the respective SMI sources specified in Core::X86::Msr::SMI_ON_IO_TRAP.



**SMMxFEC8 [IO Restart Byte] (Core::X86::Smm::IoRestart)**

Read-write. Reset: 00h.

If the core entered SMM on an IO instruction boundary, the SMI handler may write this to FFh. This causes the core to re-execute the trapped IO instruction immediately after resuming from SMM. The SMI handler should only write to this byte if Core::X86::Smm::TrapOffset[V] == 1; otherwise, the behavior is undefined.

If a second SMI is asserted while a valid IO instruction is trapped by the first SMI handler, the core services the second SMI prior to re-executing the trapped IO instruction. Core::X86::Smm::TrapOffset[V] == 0 during the second entry into SMM, and the second SMI handler must not rewrite this byte.

If there is a simultaneous SMI IO instruction trap and debug breakpoint trap, the processor first responds to the SMI and postpones recognizing the debug exception until after resuming from SMM. If debug registers other than DR6 and DR7 are used while in SMM, they must be saved and restored by the SMI handler. If Core::X86::Smm::IoRestart is set to FFh when the RSM instruction is executed, the debug trap does not occur until after the IO instruction is re-executed.

Bits	Description
7:0	<b>RST: SMM IO Restart Byte.</b> Read-write. Reset: 00h.

**SMMxFEC9 [Auto Halt Restart Offset] (Core::X86::Smm::AutoHalt)**

Read-write. Reset: 00h.

Bits	Description
7:1	Reserved.
0	<b>HLT: halt restart.</b> Read-write. Reset: 0. 0=Entered SMM on a normal x86 instruction boundary. 1=Entered SMM from the Halt state. Upon SMM entry, this bit indicates whether SMM was entered from the Halt state. Before returning from SMM, this bit can be written by the SMI handler to specify whether the return from SMM should take the processor back to the Halt state or to the instruction-execution state specified by the SMM state save area (normally, the instruction after the halt). Clearing this bit the returns to the instruction specified in the SMM save state. Setting this bit returns to the halt state. If the return from SMM takes the processor back to the Halt state, the HLT instruction is not refetched and re-executed. However, the Halt special bus cycle is broadcast and the processor enters the Halt state.

**SMMxFECA [NMI Mask] (Core::X86::Smm::NmiMask)**

Read-write. Reset: 00h.

Bits	Description
7:1	Reserved.
0	<b>NmiMask: NMI Mask.</b> Read-write. Reset: 0. 0=NMI not masked. 1=NMI masked. Specifies whether NMI was masked upon entry to SMM.

SMMxFED8 [SMM SVM State] (Core::X86::Smm::SvmState)	
Read-only, Volatile. Reset: 0000_0000_0000_0000h.	
This offset stores the SVM state of the processor upon entry into SMM.	
Bits	Description
63:4	Reserved.
3	<b>HostEflagsIF</b> : host EFLAGS IF. Read-only, Volatile. Reset: 0.
2:0	<b>SvmState</b> . Read-only, Volatile. Reset: 0h.
<b>Valid Values:</b>	
Value	Description
0h	SMM entered from a non-guest state.
1h	Reserved.
2h	SMM entered from a guest state.
5h-3h	Reserved.
6h	SMM entered from a guest state with nested paging enabled.
7h	Reserved.

SMMxFEFC [SMM Revision Identifier] (Core::X86::Smm::SmmRevID)	
Read-only. Reset: 0003_0064h.	
This offset stores the SVM state of the processor upon entry into SMM.	
Bits	Description
31:18	Reserved.
17	<b>BRL</b> . Read-only. Reset: 1. 1=Base relocation supported.
16	<b>IOTrap</b> . Read-only. Reset: 1. 1=IO trap supported.
15:0	<b>Revision</b> . Read-only. Reset: 0064h.

SMMxFE00 [SMM Base Address] (Core::X86::Smm::SmmBase)	
Read-write, Volatile. Reset: 0000_0000_0000_0000h.	
This offset stores the base of the SMM-State of the processor upon entry into SMM.	
Bits	Description
63:32	Reserved.
31:0	<b>SmmBase</b> . Read-write, Volatile. Reset: 0000_0000h. See Core::X86::Msrr::SMM_BASE[SmmBase].

#### 2.1.11.1.7 Exceptions and Interrupts in SMM

When SMM is entered, the core masks INTR, NMI, SMI, and INIT interrupts. The core clears the IF flag to disable INTR interrupts. To enable INTR interrupts within SMM, the SMM handler must set the IF flag to 1.

Generating an INTR interrupt can be used for unmasking NMI interrupts in SMM. The core recognizes the assertion of NMI within SMM immediately after the completion of an IRET instruction. Once NMI is recognized within SMM, NMI recognition remains enabled until SMM is exited, at which point NMI masking is restored to the state it was in before entering SMM.

While in SMM, the core responds to STPCLK interrupts, as well as to all exceptions that may be caused by the SMI handler.

#### 2.1.11.1.8 The Protected ASeg and TSeg Areas

These ranges are controlled by Core::X86::Msrr::SMMAddr and Core::X86::Msrr::SMMMask; see those registers for details.

### 2.1.11.1.9 SMM Special Cycles

Special cycles can be initiated on entry and exit from SMM to acknowledge to the system that these transitions are occurring. These are controlled by Core::X86::Msr::HWCR[RsmSpCycDis,SmiSpCycDis].

### 2.1.11.1.10 Locking SMM

The SMM registers (Core::X86::Msr::SMMAddr and Core::X86::Msr::SMMMask) can be locked from being altered by setting Core::X86::Msr::HWCR[SmmLock]. SBIOS must lock the SMM registers after initialization to prevent unexpected changes to these registers.

### 2.1.11.1.11 SMM Page Configuration Lock

The SMM Page Configuration Lock feature allows SMM handler code to lock the paging configuration. Once locked, the paging configuration cannot be modified until RSM completes.

Core::X86::Cpuid::FeatureExt2Eax[SmmPgCfgLock] Specifies SMM page configuration locking is supported.

Core::X86::Msr::HWCR[SmmPgCfgLock] locks registers related to page configuration. If not in SMM mode, Error-on-write-1. Cleared on RSM instruction.

If Core::X86::Msr::HWCR[SmmPgCfgLock], WRMSR of Core::X86::Msr::EFER results in an error.

If Core::X86::Msr::HWCR[SmmPgCfgLock], MOV CR0, CR3 and CR4 instructions result in an error.

### 2.1.11.2 Local APIC

The processor supports the APIC interrupt controller and the X2APIC interrupt controllers.

See 2.1.11.2.2 [Local APIC Registers] for the APIC registers and Core::X86::Msr::APIC\_ID through Core::X86::Msr::ExtendedInterruptLvtEntries for the X2APIC registers.

#### 2.1.11.2.1 Local APIC Functional Description

The local APIC contains logic to receive interrupts from a variety of sources and to send interrupts to other local APICs, as well as registers to control its behavior and report status. Interrupts can be received from:

- IO devices including the IO hub (IO APICs)
- Other local APICs (inter-processor interrupts)
- APIC timer
- Thermal events
- Performance counters
- Legacy local interrupts from the IO hub (INTR and NMI)
- APIC internal errors

The APIC timer, thermal events, performance counters, local interrupts, and internal errors are all considered local interrupt sources, and their routing is controlled by local vector table entries. These entries assign a message type and vector to each interrupt, allow them to be masked, and track the status of the interrupt.

IO and inter-processor interrupts have their message type and vector assigned at the source and are unaltered by the local APIC. They carry a destination field and a mode bit that together determine which local APIC(s) accepts them. The destination mode (DM) bit specifies if the interrupt request packet should be handled in physical or logical destination mode.

#### 2.1.11.2.1.1 Detecting and Enabling

The presence of APIC is detected via Core::X86::Cpuid::FeatureIdEdx[APIC], and the presence of X2APIC is detected

via Core::X86::Cpuid::FeatureIdEcX[X2APIC].

The local APIC is enabled via Core::X86::Msr::APIC\_BAR[ApicEn]. The X2APIC is enabled via Core::X86::Msr::APIC\_BAR[x2ApicEn]. Reset forces the APIC and X2APIC disabled.

### 2.1.11.2.1.2 APIC Register Space

MMIO APIC space:

- Memory mapped to a 4-KB range. The memory type of this space is the UC memory type. The base address of this range is specified by {Core::X86::Msr::APIC\_BAR[ApicBar[47:12]],000h}.
- The mnemonic is defined to be APICxXXX; where XXX is the byte address offset from the base address starting with APICx020 through APICx530 (Core::X86::Apic::ApicId - Core::X86::Apic::ExtendedInterruptLvtEntries).
- Treated as normal memory space when APIC is disabled, as specified by Core::X86::Msr::APIC\_BAR[ApicEn].

MSR X2APIC space:

- The local APIC register space in x2APIC mode.
- MMIO APIC registers in x2APIC mode is defined by the register from MSR0000\_0802 to MSR0000\_08[53:50] (Core::X86::Msr::APIC\_ID through Core::X86::Msr::ExtendedInterruptLvtEntries).
- If (Core::X86::Msr::APIC\_BAR[x2ApicEn] == 0) then GP-read-write.
- RDMSR/WRMSR will occur in program order.

### 2.1.11.2.1.3 ApicId Enumeration Requirements

Note: do not require contiguous ApicId assignments.

Operating systems are expected to use Core::X86::Cpuid::SizeId[ApicIdSize], the number of least significant bits in the Initial APIC ID that indicate core ID within a processor, in constructing per-core CPUID masks. Core::X86::Cpuid::SizeId[ApicIdSize] determines the maximum number of cores (MNC) that the processor could theoretically support, not the actual number of cores that are actually implemented or enabled on the processor, as indicated by Core::X86::Cpuid::SizeId[NC].

### 2.1.11.2.1.4 Physical Destination Mode

The interrupt is only accepted by the local APIC whose Core::X86::Apic::ApicId[ApicId] matches the destination field of the interrupt. Physical mode allows up to 255 APICs to be addressed individually.

### 2.1.11.2.1.5 Logical Destination Mode

A local APIC accepts interrupts selected by Core::X86::Apic::LocalDestination and the destination field of the interrupt using either cluster or flat format as configured by Core::X86::Apic::DestinationFormat[Format].

If flat destinations are in use, bits[7:0] of Core::X86::Apic::LocalDestination[Destination] are checked against bits[7:0] of the arriving interrupt's destination field. If any bit position is set in both fields, the local APIC is a valid destination. Flat format allows up to 8 APICs to be addressed individually.

If cluster destinations are in use, bits[7:4] of Core::X86::Apic::LocalDestination[Destination] are checked against bits[7:4] of the arriving interrupt's destination field to identify the cluster. If all of bits[7:4] match, then bits[3:0] of Core::X86::Apic::LocalDestination[Destination] and the interrupt destination are checked for any bit positions that are set in both fields to identify processors within the cluster. If both conditions are met, the local APIC is a valid destination. Cluster format allows 15 clusters of 4 APICs each to be addressed.

**2.1.11.2.1.6 Interrupt Delivery**

SMI, NMI, INIT, Startup, and External interrupts are classified as non-vectored interrupts.

When an APIC accepts a non-vectored interrupt, it is handled directly by the processor instead of being queued in the APIC. When an APIC accepts a fixed or lowest-priority interrupt, it sets the bit in Core::X86::Apic::InterruptRequest corresponding to the vector in the interrupt. For local interrupt sources, this comes from the vector field in that interrupt's local vector table entry. The corresponding bit in Core::X86::Apic::TriggerMode is set if the interrupt is level-triggered and cleared if edge-triggered. If a subsequent interrupt with the same vector arrives when the corresponding bit in Core::X86::Apic::InterruptRequest[RequestBits] is already set, the two interrupts are collapsed into one. Vectors[15:0] are Reserved.

**2.1.11.2.1.7 Vectored Interrupt Handling**

Core::X86::Apic::TaskPriority and Core::X86::Apic::ProcessorPriority each contain an 8-bit priority divided into a main priority (bits[7:4]) and a priority sub-class (bits[3:0]). The task priority is assigned by software to set a threshold priority at which the processor is interrupted.

The processor priority is calculated by comparing the main priority (bits[7:4]) of Core::X86::Apic::TaskPriority[Priority] to bits[7:4] of the 8-bit encoded value of the highest bit set in Core::X86::Apic::InService. The processor priority is the higher of the two main priorities.

The processor priority is used to determine if any accepted interrupts (indicated by Core::X86::Apic::InterruptRequest[RequestBits]) are high enough priority to be serviced by the processor. When the processor is ready to service an interrupt, the highest bit in Core::X86::Apic::InterruptRequest[RequestBits] is cleared, and the corresponding bit is set in Core::X86::Apic::InService[InServiceBits].

When the processor has completed service for an interrupt, it performs a Write to Core::X86::Apic::EndOfInterrupt, clearing the highest bit in Core::X86::Apic::InService[InServiceBits] and causing the next-highest interrupt to be serviced. If the corresponding bit in Core::X86::Apic::TriggerMode[TriggerModeBits] is set, a Write to Core::X86::Apic::EndOfInterrupt is performed on all APICs to complete service of the interrupt at the source.

**2.1.11.2.1.8 Interrupt Masking**

Interrupt masking is controlled by the Core::X86::Apic::ExtendedApicControl. If Core::X86::Apic::ExtendedApicControl[IerEn] is set, Core::X86::Apic::InterruptEnable are used to mask interrupts. Any bit in Core::X86::Apic::InterruptEnable[InterruptEnableBits] that is clear indicates the corresponding interrupt is masked. A masked interrupt is not serviced and the corresponding bit in Core::X86::Apic::InterruptRequest[RequestBits] remains set.

**2.1.11.2.1.9 Spurious Interrupts**

In the event that the task priority is set to or above the level of the interrupt to be serviced, the local APIC delivers a spurious interrupt vector to the processor, as specified by Core::X86::Apic::SpuriousInterruptVector. Core::X86::Apic::InService is not changed and no Write to Core::X86::Apic::EndOfInterrupt occurs.

**2.1.11.2.1.10 Spurious Interrupts Caused by Timer Tick Interrupt**

A typical interrupt is asserted until it is serviced. An interrupt is de-asserted when software clears the interrupt status bit within the interrupt service routine. Timer tick interrupt is an exception since it is de-asserted regardless of whether it is serviced or not.

The processor is not always able to service interrupts immediately (i.e., when interrupts are masked by clearing EFLAGS.IM).

If the processor is not able to service the timer tick interrupt for an extended period of time, the INTR caused by the first timer tick interrupt asserted during that time is delivered to the local APIC in ExtInt mode and latched, and the subsequent timer tick interrupts are lost. The following cases are possible when the processor is ready to service interrupts:

- An ExtInt interrupt is pending, and INTR is asserted. This results in timer tick interrupt servicing. This occurs 50 percent of the time.
- An ExtInt interrupt is pending, and INTR is de-asserted. The processor sends the interrupt acknowledge cycle, but when the PIC receives it, INTR is de-asserted, and the PIC sends a spurious interrupt vector. This occurs 50 percent of the time.

There is a 50 percent probability of spurious interrupts to the processor.

### 2.1.11.2.1.11 Lowest-Priority Interrupt Arbitration

Fixed and non-vectored interrupts are accepted by their destination APICs without arbitration.

Delivery of lowest-priority interrupts requires all APICs to arbitrate to determine which one accepts the interrupt. If Core::X86::Apic::SpuriousInterruptVector[FocusDisable] is clear, then the focus processor for an interrupt always accepts the interrupt. A processor is the focus of an interrupt if it is already servicing that interrupt (corresponding bit in Core::X86::Apic::InService[InServiceBits] is set) or if it already has a pending request for that interrupt (corresponding bit in Core::X86::Apic::InterruptRequest[RequestBits] is set). If Core::X86::Apic::ExtendedApicControl[IerEn] is set, the interrupt must also be enabled in Core::X86::Apic::InterruptEnable[InterruptEnableBits] for a processor to be the focus processor. If there is no focus processor for an interrupt, or focus processor checking is disabled, then each APIC calculates an arbitration priority value, stored in Core::X86::Apic::ArbitrationPriority, and the one with the lowest result accepts the interrupt.

The arbitration priority value is calculated by comparing Core::X86::Apic::TaskPriority[Priority] with the 8-bit encoded value of the highest bit set in Core::X86::Apic::InterruptRequest[RequestBits] (IRRVec) and the 8-bit encoded value of the highest bit set Core::X86::Apic::InService[InServiceBits] (ISRVec). If Core::X86::Apic::ExtendedApicControl[IerEn] is set the IRRVec and ISRVec are based off the highest enabled interrupt. The main priority bits[7:4] are compared as follows:

```
if ((TaskPriority[Priority[7:4]] >= InterruptRequest[IRRVec[7:4]])
&&(TaskPriority[Priority[7:4]] > InService[ISRVec[7:4]]) {
ArbitrationPriority[Priority] = TaskPriority[Priority]
} elseif { (InterruptRequest[IRRVec[7:4]] > InService[ISRVec[7:4]])
ArbitrationPriority[Priority] = {InterruptRequest[IRRVec[7:4]], 0h}
} else {
ArbitrationPriority[Priority] = {InService[ISRVec[7:4]], 0h}
}
```

### 2.1.11.2.1.12 Inter-Processor Interrupts

The Core::X86::Apic::InterruptCommandLow and Core::X86::Apic::InterruptCommandHigh provide a mechanism for generating interrupts in order to redirect an interrupt to another processor, originate an interrupt to another processor, or allow a processor to interrupt itself. A Write to register Core::X86::Apic::InterruptCommandLow causes an interrupt to be generated with the properties specified by the Core::X86::Apic::InterruptCommandLow and Core::X86::Apic::InterruptCommandHigh fields.

Not all combinations of ICR fields are valid. Only the following combinations are valid:

Note: x indicates a don't care.

*Table 17: ICR Valid Combinations*

Message Type	Trigger Mode	Level	Destination Shorthand
Fixed	Edge	x	x
	Level	Assert	x
Lowest Priority, SMI, NMI, INIT	Edge	x	Destination or all excluding self
	Level	Assert	Destination or all excluding self
Startup	x	x	Destination or all excluding self

#### 2.1.11.2.1.13 APIC Timer Operation

The local APIC contains a 32-bit timer, controlled by Core::X86::Apic::TimerLvtEntry, Core::X86::Apic::TimerInitialCount, and Core::X86::Apic::TimerDivideConfiguration. The processor bus clock is divided by the value in Core::X86::Apic::TimerDivideConfiguration[Div[3:0]] to obtain a time base for the timer. When Core::X86::Apic::TimerInitialCount[Count] is written, the value is copied into Core::X86::Apic::TimerCurrentCount. Core::X86::Apic::TimerCurrentCount[Count] is decremented at the rate of the divided clock. When the count reaches 0, a timer interrupt is generated with the vector specified in Core::X86::Apic::TimerLvtEntry[Vector]. If Core::X86::Apic::TimerLvtEntry[Mode] specifies periodic operation, Core::X86::Apic::TimerCurrentCount[Count] is reloaded with the Core::X86::Apic::TimerInitialCount[Count] value, and it continues to decrement at the rate of the divided clock. If Core::X86::Apic::TimerLvtEntry[Mask] is set, timer interrupts are not generated.

#### 2.1.11.2.1.14 Generalized Local Vector Table

All LVTs (Core::X86::Apic::ThermalLvtEntry to Core::X86::Apic::LVTINT, and Core::X86::Apic::ExtendedInterruptLvtEntries) support a generalized message type as follows:

- 000b=Fixed
- 010b=SMI
- 100b=NMI
- 111b=ExtINT
- All other messages types are Reserved.

#### 2.1.11.2.1.15 State at Reset

At power-up or reset, the APIC is hardware disabled (Core::X86::Msr::APIC\_BAR[ApicEn] == 0) so only SMI, NMI, INIT, and ExtInt interrupts may be accepted.

The APIC can be software disabled through Core::X86::Apic::SpuriousInterruptVector[APICSWEn]. The software disable has no effect when the APIC is hardware disabled.

When a processor accepts an INIT interrupt, the APIC is reset as at power-up, with the exception that:

- Core::X86::Apic::ApicId is unaffected.
- Pending APIC register writes complete.



### 2.1.11.2.2 Local APIC Registers

APICx020 [APIC ID] (Core::X86::Apic::ApicId)	
Read-only.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; APICx020; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}	
Bits	Description
31:24	<b>ApicId: APIC ID.</b> Read-only. Reset: XXh. The reset value varies based on core number. See 2.1.11.2.1.3 [ApicId Enumeration Requirements].
23:0	Reserved.
APICx030 [APIC Version] (Core::X86::Apic::ApicVersion)	
Read-only.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; APICx030; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}	
Bits	Description
31	<b>ExtApicSpace: extended APIC register space present.</b> Read-only. Reset: 1. 1=Indicates the presence of extended APIC register space starting at Core::X86::Apic::ExtendedApicFeature.
30:25	Reserved.
24	<b>DirectedEoiSupport: directed EOI support.</b> Read-only. Reset: Fixed,1. 0=Directed EOI capability not supported.
23:16	<b>MaxLvtEntry.</b> Read-only. Reset: XXh. Specifies the number of entries in the local vector table minus one.
15:8	Reserved.
7:0	<b>Versioin.</b> Read-only. Reset: 10h. Indicates the version number of this APIC implementation.
APICx080 [Task Priority] (Core::X86::Apic::TaskPriority)	
Read-write. Reset: 0000_0000h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; APICx080; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}	
Bits	Description
31:8	Reserved.
7:0	<b>Priority.</b> Read-write. Reset: 00h. This field is assigned by software to set a threshold priority at which the core is interrupted.
APICx090 [Arbitration Priority] (Core::X86::Apic::ArbitrationPriority)	
Read-only, Volatile. Reset: 0000_0000h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; APICx090; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}	
Bits	Description
31:8	Reserved.
7:0	<b>Priority.</b> Read-only, Volatile. Reset: 00h. Indicates the current priority for a pending interrupt, or a task or interrupt being serviced by the core. The priority is used to arbitrate between cores to determine which accepts a lowest-priority interrupt request.
APICx0A0 [Processor Priority] (Core::X86::Apic::ProcessorPriority)	
Read-only, Volatile. Reset: 0000_0000h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; APICx0A0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}	
Bits	Description
31:8	Reserved.
7:0	<b>Priority.</b> Read-only, Volatile. Reset: 00h. Indicates the core's current priority servicing a task or interrupt, and is used to determine if any pending interrupts should be serviced. It is the higher value of the task priority value and the current highest in-service interrupt.



**APICx0B0 [End of Interrupt] (Core::X86::Apic::EndOfInterrupt)**

Write-only.

This register is written by the software interrupt handler to indicate the servicing of the current interrupt is complete.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx0B0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

**Bits Description**

31:0 Reserved.

**APICx0C0 [Reserved] (Core::X86::Apic::RemoteRead)**

Read-only. Reset: 0000\_0000h.

Remote Read is not supported.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx0C0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

**Bits Description**

31:0 Reserved.

**APICx0D0 [Logical Destination] (Core::X86::Apic::LocalDestination)**

Read-write, Volatile. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx0D0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

**Bits Description**31:24 **Destination.** Read-write, Volatile. Reset: 00h. This APIC's destination identification. Used to determine which interrupts should be accepted.

23:0 Reserved.

**APICx0E0 [Destination Format] (Core::X86::Apic::DestinationFormat)**

Read-write. Reset: F000\_0000h.

Only supported in xAPIC mode.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx0E0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

**Bits Description**31:28 **Format.** Read-write. Reset: Fh. Controls which format to use when accepting interrupts with a logical destination mode.**Valid Values:**

Value	Description
0h	Cluster destinations are used.
Eh-1h	Reserved.
Fh	Flat destinations are used.

27:0 Reserved.

**APICx0F0 [Spurious-Interrupt Vector] (Core::X86::Apic::SpuriousInterruptVector)**

Reset: 0000\_00FFh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx0F0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

**Bits Description**

31:10 Reserved.

9 **FocusDisable.** Read-write. Reset: 0. 1=Disable focus core checking during lowest-priority arbitrated interrupts.8 **APICSWEn: APIC software enable.** Read-write, Volatile. Reset: 0. 0=SMI, NMI, INIT, LINT[1:0], and Startup interrupts may be accepted; pending interrupts in Core::X86::Apic::InService and Core::X86::Apic::InterruptRequest are held, but further fixed, lowest-priority, and ExtInt interrupts are not accepted. All LVT entry mask bits are set and cannot be cleared.7:0 **Vector.** Read-write, Volatile. Reset: FFh. The vector that is sent to the core in the event of a spurious interrupt.

**APICx1[0...7]0 [In-Service] (Core::X86::Apic::InService)**

Read-only, Volatile. Reset: 0000\_0000h.

The in-service registers provide a bit per interrupt to indicate that the corresponding interrupt is being serviced by the core. The first 16 InServiceBits of the first Core::X86::Apic::InService register are Reserved.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; APICx100; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; APICx110; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; APICx120; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; APICx130; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n4; APICx140; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n5; APICx150; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n6; APICx160; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n7; APICx170; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	<b>InServiceBits.</b> Read-only, Volatile. Reset: 0000_0000h. These bits are set when the corresponding interrupt is being serviced by the core.

**APICx1[8...F]0 [Trigger Mode] (Core::X86::Apic::TriggerMode)**

Read-only, Volatile. Reset: 0000\_0000h.

The trigger mode registers provide a bit per interrupt to indicate the assertion mode of each interrupt. The first 16 TriggerModeBits of the each thread's APIC[1F0:180] registers are Reserved.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; APICx180; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; APICx190; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; APICx1A0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; APICx1B0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n4; APICx1C0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n5; APICx1D0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n6; APICx1E0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n7; APICx1F0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	<b>TriggerModeBits.</b> Read-only, Volatile. Reset: 0000_0000h. The corresponding trigger mode bit is updated when an interrupt is accepted. 1=Level-triggered interrupt. 0=Edge-triggered interrupt. <b>ValidValues:</b> See: Core::X86::Apic::TriggerMode[TriggerModeBits] Valid Values.

**Core::X86::Apic::TriggerMode[TriggerModeBits] Valid Values**

Bit	Description
[0]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[1]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[2]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[3]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[4]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[5]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[6]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[7]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[8]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[9]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[10]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[11]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[12]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[13]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[14]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[15]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[16]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[17]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.

[18]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[19]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[20]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[21]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[22]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[23]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[24]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[25]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[26]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[27]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[28]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[29]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[30]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.
[31]	0=Edge-triggered interrupt. 1=Level-triggered interrupt.

#### APICx2[0...7]0 [Interrupt Request] (Core::X86::Apic::InterruptRequest)

Read-only. Reset: 0000\_0000h.

The interrupt request registers provide a bit per interrupt to indicate that the corresponding interrupt has been accepted by the APIC. The first 16 RequestBits of the first Core::X86::Apic::InterruptRequest register are Reserved.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; APICx200; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; APICx210; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; APICx220; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; APICx230; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n4; APICx240; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n5; APICx250; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n6; APICx260; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n7; APICx270; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	<b>RequestBits.</b> Read-only. Reset: 0000_0000h. The corresponding request bit is set when the an interrupt is accepted by the APIC. <b>ValidValues:</b> See: Core::X86::Apic::InterruptRequest[RequestBits] Valid Values.

#### Core::X86::Apic::InterruptRequest[RequestBits] Valid Values

Bit	Description
[0]	0=Request bit not set. 1=Request bit set.
[1]	0=Request bit not set. 1=Request bit set.
[2]	0=Request bit not set. 1=Request bit set.
[3]	0=Request bit not set. 1=Request bit set.
[4]	0=Request bit not set. 1=Request bit set.
[5]	0=Request bit not set. 1=Request bit set.
[6]	0=Request bit not set. 1=Request bit set.
[7]	0=Request bit not set. 1=Request bit set.
[8]	0=Request bit not set. 1=Request bit set.
[9]	0=Request bit not set. 1=Request bit set.
[10]	0=Request bit not set. 1=Request bit set.
[11]	0=Request bit not set. 1=Request bit set.
[12]	0=Request bit not set. 1=Request bit set.
[13]	0=Request bit not set. 1=Request bit set.
[14]	0=Request bit not set. 1=Request bit set.
[15]	0=Request bit not set. 1=Request bit set.
[16]	0=Request bit not set. 1=Request bit set.

[17]	0=Request bit not set. 1=Request bit set.
[18]	0=Request bit not set. 1=Request bit set.
[19]	0=Request bit not set. 1=Request bit set.
[20]	0=Request bit not set. 1=Request bit set.
[21]	0=Request bit not set. 1=Request bit set.
[22]	0=Request bit not set. 1=Request bit set.
[23]	0=Request bit not set. 1=Request bit set.
[24]	0=Request bit not set. 1=Request bit set.
[25]	0=Request bit not set. 1=Request bit set.
[26]	0=Request bit not set. 1=Request bit set.
[27]	0=Request bit not set. 1=Request bit set.
[28]	0=Request bit not set. 1=Request bit set.
[29]	0=Request bit not set. 1=Request bit set.
[30]	0=Request bit not set. 1=Request bit set.
[31]	0=Request bit not set. 1=Request bit set.

#### APICx280 [Error Status] (Core::X86::Apic::ErrorStatus)

Writes to this register trigger an update of the register state. The value written by software is arbitrary. Each write causes the internal error state to be loaded into this register, clearing the internal error state. Consequently, a second write prior to the occurrence of another error causes the register to be overwritten with cleared data.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx280; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]], 000h}

Bits	Description
31:8	Reserved.
7	<b>IllegalRegAddr: illegal register address.</b> Read-write. Reset: 0. This bit indicates that an access to a nonexistent register location within this APIC was attempted. Can only be set in xAPIC mode.
6	<b>RcvdIllegalVector: received illegal vector.</b> Read-write. Reset: 0. This bit indicates that this APIC has received a message with an illegal vector (00h to 0Fh for fixed and lowest priority interrupts).
5	<b>SentIllegalVector.</b> Read-write. Reset: 0. This bit indicates that this APIC attempted to send a message with an illegal vector (00h to 0Fh for fixed and lowest priority interrupts).
4	Reserved.
3	<b>RcvAcceptError: receive accept error.</b> Read-write. Reset: 0. This bit indicates that a message received by this APIC was not accepted by this or any other APIC.
2	<b>SendAcceptError.</b> Read-write. Reset: 0. This bit indicates that a message sent by this APIC was not accepted by any APIC.
1:0	Reserved.

**APICx300 [Interrupt Command Low] (Core::X86::Apic::InterruptCommandLow)**

Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx300; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description																		
31:20	Reserved.																		
19:18	<b>DestShrthnd: destination shorthand.</b> Read-write. Reset: 0h. <b>Description:</b> Provides a quick way to specify a destination for a message. If all including self or all excluding self is used, then destination mode is ignored and physical is automatically used. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>No shorthand (Destination field).</td></tr> <tr> <td>1h</td><td>Self.</td></tr> <tr> <td>2h</td><td>All including self.</td></tr> <tr> <td>3h</td><td>All excluding self (This sends a message with a destination encoding of all 1s, so if lowest priority is used the message could end up being reflected back to this APIC).</td></tr> </table>	Value	Description	0h	No shorthand (Destination field).	1h	Self.	2h	All including self.	3h	All excluding self (This sends a message with a destination encoding of all 1s, so if lowest priority is used the message could end up being reflected back to this APIC).								
Value	Description																		
0h	No shorthand (Destination field).																		
1h	Self.																		
2h	All including self.																		
3h	All excluding self (This sends a message with a destination encoding of all 1s, so if lowest priority is used the message could end up being reflected back to this APIC).																		
17:16	<b>RemoteRdStat.</b> Read-only. Reset: 0h. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Read was invalid.</td></tr> <tr> <td>1h</td><td>Delivery pending.</td></tr> <tr> <td>2h</td><td>Delivery complete and access was valid.</td></tr> <tr> <td>3h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	Read was invalid.	1h	Delivery pending.	2h	Delivery complete and access was valid.	3h	Reserved.								
Value	Description																		
0h	Read was invalid.																		
1h	Delivery pending.																		
2h	Delivery complete and access was valid.																		
3h	Reserved.																		
15	<b>TM: trigger mode.</b> Read-write. Reset: 0. 0=Edge triggered. 1=Level triggered. Indicates how this interrupt is triggered.																		
14	<b>Level.</b> Read-write. Reset: 0. 0=De-asserted. 1=Asserted.																		
13	Reserved.																		
12	<b>DS: interrupt delivery status.</b> Read-only. Reset: 0. 0=Idle. 1=Send pending. In xAPIC mode this bit is set to indicate that the interrupt has not yet been accepted by the destination core(s). Software may repeatedly write Core::X86::Apic::InterruptCommandLow without polling the DS bit; all requested IPIs are delivered.																		
11	<b>DM: destination mode.</b> Read-write. Reset: 0. 0=Physical. 1=Logical.																		
10:8	<b>MsgType.</b> Read-write. Reset: 0h. The message types are encoded as follows: <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Fixed.</td></tr> <tr> <td>1h</td><td>Lowest Priority.</td></tr> <tr> <td>2h</td><td>SMI.</td></tr> <tr> <td>3h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>NMI.</td></tr> <tr> <td>5h</td><td>INIT.</td></tr> <tr> <td>6h</td><td>Startup.</td></tr> <tr> <td>7h</td><td>External interrupt.</td></tr> </table>	Value	Description	0h	Fixed.	1h	Lowest Priority.	2h	SMI.	3h	Reserved.	4h	NMI.	5h	INIT.	6h	Startup.	7h	External interrupt.
Value	Description																		
0h	Fixed.																		
1h	Lowest Priority.																		
2h	SMI.																		
3h	Reserved.																		
4h	NMI.																		
5h	INIT.																		
6h	Startup.																		
7h	External interrupt.																		
7:0	<b>Vector.</b> Read-write. Reset: 00h. The vector that is sent for this interrupt source.																		

**APICx310 [Interrupt Command High] (Core::X86::Apic::InterruptCommandHigh)**

Read-write. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx310; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:24	<b>DestinationField.</b> Read-write. Reset: 00h. The destination encoding used when Core::X86::Apic::InterruptCommandLow[DestShrthnd] is 00b.
23:0	Reserved.

**APICx320 [LVT Timer] (Core::X86::Apic::TimerLvtEntry)**

Reset: 0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx320; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:18	Reserved.
17	<b>Mode.</b> Read-write. Reset: 0. 0=One-shot. 1=Periodic.
16	<b>Mask.</b> Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Read-only, Volatile. Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.)
11	Reserved.
10:8	<b>MsgType: message type.</b> Read-write. Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table].
7:0	<b>Vector.</b> Read-write. Reset: 00h. Interrupt vector number.

**APICx330 [LVT Thermal Sensor] (Core::X86::Apic::ThermalLvtEntry)**

Reset: 0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx330; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	<b>Mask.</b> Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Read-only, Volatile. Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.)
11	Reserved.
10:8	<b>MsgType: message type.</b> Read-write. Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table].
7:0	<b>Vector.</b> Read-write. Reset: 00h. Interrupt vector number.

**APICx340 [LVT Performance Monitor] (Core::X86::Apic::PerformanceCounterLvtEntry)**

Reset: 0001\_0000h.

Interrupts for this local vector table are caused by overflows of:

- Core::X86::Msr::PERF\_LEGACY\_CTL0..3(Performance Event Select [3:0]).
- Core::X86::Msr::PERF\_CTL0..5(Performance Event Select [5:0]).

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx340; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	<b>Mask.</b> Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Read-only, Volatile. Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.)
11	Reserved.
10:8	<b>MsgType: message type.</b> Read-write. Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table].
7:0	<b>Vector.</b> Read-write. Reset: 00h. Interrupt vector number.

**APICx3[5...6]0 [LVT LINT[1:0]] (Core::X86::Apic::LVTINT)**

Reset: 0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; APICx350; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; APICx360; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	<b>Mask.</b> Read-write. Reset: 1. 0=Not masked. 1=Masked.
15	<b>TM: trigger mode.</b> Read-write. Reset: 0. 0=Edge. 1=Level.
14	<b>RmtIRR.</b> Read-only, Volatile. Reset: 0. If trigger mode is level, remote Core::X86::Apic::InterruptRequest is set when the interrupt has begun service. Remote Core::X86::Apic::InterruptRequest is cleared when the end of interrupt has occurred.
13	Reserved.
12	<b>DS: interrupt delivery status.</b> Read-only, Volatile. Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.)
11	Reserved.
10:8	<b>MsgType: message type.</b> Read-write. Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table].
7:0	<b>Vector.</b> Read-write. Reset: 00h. Interrupt vector number.

**APICx370 [LVT Error] (Core::X86::Apic::ErrorLvtEntry)**

Reset: 0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx370; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	<b>Mask.</b> Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Read-only, Volatile. Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.)
11	Reserved.
10:8	<b>MsgType: message type.</b> Read-write. Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table].
7:0	<b>Vector.</b> Read-write. Reset: 00h. Interrupt vector number.

**APICx380 [Timer Initial Count] (Core::X86::Apic::TimerInitialCount)**

Read-write, Volatile. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx380; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	<b>Count.</b> Read-write, Volatile. Reset: 0000_0000h. The value copied into the current count register when the timer is loaded or reloaded.

**APICx390 [Timer Current Count] (Core::X86::Apic::TimerCurrentCount)**

Read-only, Volatile. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx390; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	<b>Count.</b> Read-only, Volatile. Reset: 0000_0000h. The current value of the counter.



**APICx3E0 [Timer Divide Configuration] (Core::X86::Apic::TimerDivideConfiguration)**

Read-write. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx3E0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:4	Reserved.
3:0	<b>Div[3:0]</b> . Read-write. Reset: 0h. Div[2] is unused.
	<b>ValidValues:</b>
Value	Description
0h	Divide by 2.
1h	Divide by 4.
2h	Divide by 8.
3h	Divide by 16.
7h-4h	Reserved.
8h	Divide by 32.
9h	Divide by 64.
Ah	Divide by 128.
Bh	Divide by 1.
Fh-Ch	Reserved.

**APICx400 [Extended APIC Feature] (Core::X86::Apic::ExtendedApicFeature)**

Read-only. Reset: 0004\_0007h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx400; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:24	Reserved.
23:16	<b>ExtLvtCount: extended local vector table count</b> . Read-only. Reset: 04h. This specifies the number of extended LVT registers (Core::X86::Apic::ExtendedInterruptLvtEntries) in the local APIC.
15:3	Reserved.
2	<b>ExtApicIdCap: extended APIC ID capable</b> . Read-only. Reset: 1. 1=The processor is capable of supporting an 8-bit APIC ID, as controlled by Core::X86::Apic::ExtendedApicControl[ExtApicIdEn].
1	<b>SeoiCap: specific end of interrupt capable</b> . Read-only. Reset: 1. 1=The Core::X86::Apic::SpecificEndOfInterrupt is present.
0	<b>IerCap: interrupt enable register capable</b> . Read-only. Reset: 1. This bit indicates that the Core::X86::Apic::InterruptEnable are present. See 2.1.11.2.1.8 [Interrupt Masking].

**APICx410 [Extended APIC Control] (Core::X86::Apic::ExtendedApicControl)**

Read-write. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; APICx410; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:3	Reserved.
2	<b>ExtApicIdEn: extended APIC ID enable</b> . Read-write. Reset: 0. 1=Enable 8-bit APIC ID; Core::X86::Apic::ApicId[ApicId] supports an 8-bit value; an interrupt broadcast in physical destination mode requires that the IntDest[7:0] == 1111_1111b (instead of XXXX_1111b); a match in physical destination mode occurs when (IntDest[7:0] == ApicId[7:0]) instead of (IntDest[3:0] == ApicId[3:0]).
1	<b>SeoiEn</b> . Read-write. Reset: 0. 1=Enable SEOI generation when a Write to Core::X86::Apic::SpecificEndOfInterrupt is received.
0	<b>IerEn</b> . Read-write. Reset: 0. 1=Enable writes to the interrupt enable registers.



**APICx420 [Specific End Of Interrupt] (Core::X86::Apic::SpecificEndOfInterrupt)**

Read-write. Reset: 0000\_0000h.

`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]`; APICx420; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:8	Reserved.
7:0	<b>EoiVec: end of interrupt vector.</b> Read-write. Reset: 00h. A Write to this field causes an end of interrupt cycle to be performed for the vector specified in this field. The behavior is undefined if no interrupt is pending for the specified interrupt vector.

**APICx4[8...F]0 [Interrupt Enable] (Core::X86::Apic::InterruptEnable)**

Read-write. Reset: FFFF\_FFFFh.

`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n0`; APICx480; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n1`; APICx490; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n2`; APICx4A0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n3`; APICx4B0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n4`; APICx4C0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n5`; APICx4D0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n6`; APICx4E0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n7`; APICx4F0; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	<b>InterruptEnableBits.</b> Read-write. Reset: FFFF_FFFFh. The interrupt enable bits can be used to enable each of the 256 interrupts.

**APICx5[0...3]0 [Extended Interrupt Local Vector Table] (Core::X86::Apic::ExtendedInterruptLvtEntries)**

Reset: 0001\_0000h.

Assignments conventions:

- APIC500 provides a local vector table entry for IBS.
- APIC510 provides a local vector table entry for error thresholding. See Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset].
- APIC520 provides a local vector table entry for Deferred errors. See MCI\_CONFIG[DeferredIntType].

`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n0`; APICx500; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n1`; APICx510; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n2`; APICx520; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}`_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n3`; APICx530; APIC={Core::X86::Msr::APIC\_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	<b>Mask.</b> Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Read-only, Volatile. Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.)
11	Reserved.
10:8	<b>MsgType: message type.</b> Read-write. Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table].
7:0	<b>Vector.</b> Read-write. Reset: 00h. Interrupt vector number.

**2.1.12 CPUID Instruction**

Processor feature capabilities and configuration information are provided through the CPUID instruction. The information is accessed by (1) selecting the CPUID function setting EAX and optionally ECX for some functions, (2) executing the CPUID instruction, and (3) reading the results in the EAX, EBX, ECX, and EDX registers. The syntax CPUID FnXXXXXXXX\_EiX[\_xYYY] refers to the function where EAX == X, and optionally ECX == Y, and the registers specified by EiX. EiX can be any single register such as {EAX, EBX, ECX, and EDX}, or a range of registers, such as E[C,B,A]X. Undefined function numbers return 0's in all 4 registers.

Unless otherwise specified, single-bit feature fields are encoded as 1=Feature is supported by the processor; 0=Feature is

not supported by the processor. CPUID functions not listed are reserved.

### 2.1.12.1 CPUID Instruction Functions

The following provides processor specific details about CPUID.

CPUID_Fn00000000_EAX [Processor Vendor and Largest Standard Function Number] (Core::X86::Cpuid::LargFuncNum)	
Read-only. Reset: Fixed,0000_0010h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; CPUID_Fn00000000_EAX	
Bits	Description
31:0	<b>LFuncStd: largest standard function.</b> Read-only. Reset: Fixed,0000_0010h. The largest CPUID standard function input value supported by the processor implementation.
CPUID_Fn00000000_EBX [Processor Vendor (ASCII Bytes [3:0])] (Core::X86::Cpuid::ProcVendEbx)	
Read-only. Reset: Fixed,6874_7541h.	
Core::X86::Cpuid::ProcVendEbx and Core::X86::Cpuid::ProcVendExtEbx return the same value.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; CPUID_Fn00000000_EBX	
Bits	Description
31:0	<b>Vendor.</b> Read-only. Reset: Fixed,6874_7541h. ASCII Bytes [3:0] ("h t u A") of the string "AuthenticAMD".
CPUID_Fn00000000_ECX [Processor Vendor (ASCII Bytes [11:8])] (Core::X86::Cpuid::ProcVendEcx)	
Read-only. Reset: Fixed,444D_4163h.	
Core::X86::Cpuid::ProcVendEcx and Core::X86::Cpuid::ProcVendExtEcx return the same value.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; CPUID_Fn00000000_ECX	
Bits	Description
31:0	<b>Vendor.</b> Read-only. Reset: Fixed,444D_4163h. ASCII Bytes [11:8] ("D M A c") of the string "AuthenticAMD".
CPUID_Fn00000000_EDX [Processor Vendor (ASCII Bytes [7:4])] (Core::X86::Cpuid::ProcVendEdx)	
Read-only. Reset: Fixed,6974_6E65h.	
Core::X86::Cpuid::ProcVendEdx and Core::X86::Cpuid::ProcVendExtEdx return the same value.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; CPUID_Fn00000000_EDX	
Bits	Description
31:0	<b>Vendor.</b> Read-only. Reset: Fixed,6974_6E65h. ASCII Bytes [7:4] ("i t n e") of the string "AuthenticAMD".

**CPUID\_Fn00000001\_EAX [Family, Model, Stepping Identifiers] (Core::X86::Cpuid::FamModStep)**

Read-only.

Core::X86::Cpuid::FamModStep and Core::X86::Cpuid::FamModStepExt return the same value.

Family: Is an 8-bit value and is defined as: Family[7:0]={0000b,BaseFamily[3:0]}+ExtendedFamily[7:0].

- E.g., If BaseFamily[3:0] == Fh and ExtendedFamily[7:0] == 08h, then Family[7:0] = 17h.

Model: Is an 8-bit value and is defined as: Model[7:0]={ExtendedModel[3:0],BaseModel[3:0]}.

- E.g., If ExtendedModel[3:0] == 1h and BaseModel[3:0] == 8h, then Model[7:0] = 18h.
- Model numbers vary with product.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000001\_EAX

Bits	Description
31:28	Reserved.
27:20	<b>ExtFamily: extended family.</b> Read-only. Reset: 0Bh. See Family above.
19:16	<b>ExtModel: extended model.</b> Read-only. Reset: 4h. See Model above.
15:12	Reserved.
11:8	<b>BaseFamily.</b> Read-only. Reset: Fh. See Family description above.
7:4	<b>BaseModel.</b> Read-only. Reset: Xh. Model numbers vary with product.
3:0	<b>Stepping.</b> Read-only. Reset: Xh. Processor stepping (revision) for a specific model.

**CPUID\_Fn00000001\_EBX [LocalApicId, LogicalProcessorCount, CLFlush] (Core::X86::Cpuid::FeatureIdEbx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000001\_EBX

Bits	Description
31:24	<b>LocalApicId.</b> Read-only. Reset: XXh. Initial local APIC physical ID.
23:16	<b>LogicalProcessorCount.</b> Read-only. Reset: Fixed,XXh. Specifies the number of threads in the processor as 0FFh&(Core::X86::Cpuid::SizeId[NC]+1).
15:8	<b>CLFlush.</b> Read-only. Reset: Fixed,08h. CLFLUSH size in quadwords.
7:0	Reserved.

**CPUID\_Fn00000001\_ECX [Feature Identifiers] (Core::X86::Cpuid::FeatureIdEcX)**

Read-only.

These values can be over-written by Core::X86::Msr::CPUID\_Features.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000001\_ECX

Bits	Description
31	Reserved.
30	<b>RDRAND</b> . Read-only. Reset: Fixed,1. RDRAND instruction support.
29	<b>F16C</b> . Read-only. Reset: Fixed,1. Half-precision convert instruction support.
28	<b>AVX</b> . Read-only. Reset: Fixed,1. AVX instruction support.
27	<b>OSXSAVE</b> . Read-only. Reset: X. 1=The OS has enabled support for XGETBV/XSETBV instructions to query processor extended states. OS enabled support for XGETBV/XSETBV.
26	<b>XSAVE</b> . Read-only. Reset: Fixed,1. 1=Support provided for the XSAVE, XRSTOR, XSETBV, and XGETBV instructions and the XFEATURE_ENABLED_MASK register. XSAVE (and related) instruction support.
25	<b>AES: AES instruction support</b> . Read-only. Reset: X. AES instruction support.
24	Reserved.
23	<b>POPCNT</b> . Read-only. Reset: Fixed,1. POPCNT instruction.
22	<b>MOVBE</b> . Read-only. Reset: Fixed,1. MOVBE instruction support.
21	<b>X2APIC</b> . Read-only. Reset: X. x2APIC capability.
20	<b>SSE42</b> . Read-only. Reset: Fixed,1. SSE4.2 instruction support.
19	<b>SSE41</b> . Read-only. Reset: Fixed,1. SSE4.1 instruction support.
18	Reserved.
17	<b>PCID</b> . Read-only. Reset: Fixed,0. Process context identifiers support.
16:14	Reserved.
13	<b>CMPXCHG16B</b> . Read-only. Reset: Fixed,1. CMPXCHG16B instruction.
12	<b>FMA</b> . Read-only. Reset: Fixed,1. FMA instruction support.
11:10	Reserved.
9	<b>SSSE3</b> . Read-only. Reset: Fixed,1. Supplemental SSE3 extensions.
8:4	Reserved.
3	<b>Monitor</b> . Read-only. Reset: !Core::X86::Msr::HWCR[MonMwaitDis]. Monitor/Mwait instructions.
2	Reserved.
1	<b>PCLMULQDQ</b> . Read-only. Reset: X. PCLMULQDQ instruction support.
0	<b>SSE3</b> . Read-only. Reset: Fixed,1. SSE3 extensions.

**CPUID\_Fn00000001\_EDX [Feature Identifiers] (Core::X86::Cpuid::FeatureIdEdx)**

Read-only.

These values can be over-written by Core::X86::Msr::CPUID\_Features.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000001\_EDX

Bits	Description
31:29	Reserved.
28	<b>HTT.</b> Read-only. Reset: Fixed,(Core::X86::Cpuid::SizeId[NC] != 0). 0=Single thread product (Core::X86::Cpuid::SizeId[NC] == 0). 1=Multi thread product (Core::X86::Cpuid::SizeId[NC] != 0). Hyper-threading technology.
27	Reserved.
26	<b>SSE2.</b> Read-only. Reset: Fixed,1. SSE2: SSE2 extensions.
25	<b>SSE.</b> Read-only. Reset: Fixed,1. SSE extensions.
24	<b>FXSR.</b> Read-only. Reset: Fixed,1. FXSAVE and FXRSTOR instructions.
23	<b>MMX.</b> Read-only. Reset: Fixed,1. MMX instructions
22:20	Reserved.
19	<b>CLFSH.</b> Read-only. Reset: Fixed,1. CLFLUSH instruction.
18	Reserved.
17	<b>PSE36.</b> Read-only. Reset: Fixed,1. Page-size extensions.
16	<b>PAT.</b> Read-only. Reset: Fixed,1. Page attribute table.
15	<b>CMOV.</b> Read-only. Reset: Fixed,1. Conditional move instructions, CMOV, FCOMI, FCMOV.
14	<b>MCA.</b> Read-only. Reset: Fixed,1. Machine check architecture, MCG_CAP.
13	<b>PGE.</b> Read-only. Reset: Fixed,1. Page global extension, CR4.PGE.
12	<b>MTRR.</b> Read-only. Reset: Fixed,1. Memory-type range registers.
11	<b>SysEnterSysExit.</b> Read-only. Reset: Fixed,1. SYSENTER and SYSEXIT instructions.
10	Reserved.
9	<b>APIC: advanced programmable interrupt controller (APIC) exists and is enabled.</b> Read-only. Reset: X. Core::X86::Msr::APIC_BAR[ApicEn].
8	<b>CMPXCHG8B.</b> Read-only. Reset: Fixed,1. CMPXCHG8B instruction.
7	<b>MCE.</b> Read-only. Reset: Fixed,1. Machine check exception, CR4.MCE.
6	<b>PAE.</b> Read-only. Reset: Fixed,1. Physical-address extensions (PAE).
5	<b>MSR.</b> Read-only. Reset: Fixed,1. AMD model-specific registers (MSRs), with RDMSR and WRMSR instructions.
4	<b>TSC.</b> Read-only. Reset: Fixed,1. Time Stamp Counter, RDTSC/RDTSCP instructions, CR4.TSD.
3	<b>PSE.</b> Read-only. Reset: Fixed,1. Page-size extensions (4 MB pages).
2	<b>DE.</b> Read-only. Reset: Fixed,1. Debugging extensions, IO breakpoints, CR4.DE.
1	<b>VME.</b> Read-only. Reset: Fixed,1. Virtual-mode enhancements.
0	<b>FPU.</b> Read-only. Reset: Fixed,1. x87 floating point unit on-chip.

**CPUID\_Fn00000005\_EAX [Monitor/MWait] (Core::X86::Cpuid::MonMWaitEax)**

Read-only. Reset: Fixed,0000\_0040h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000005\_EAX

Bits	Description
31:16	Reserved.
15:0	<b>MonLineSizeMin.</b> Read-only. Reset: Fixed,0040h. Smallest monitor-line size in bytes.

**CPUID\_Fn00000005\_EBX [Monitor/MWait] (Core::X86::Cpuid::MonMWaitEbx)**

Read-only. Reset: Fixed,0000\_0040h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000005\_EBX

Bits	Description
31:16	Reserved.
15:0	<b>MonLineSizeMax</b> . Read-only. Reset: Fixed,0040h. Largest monitor-line size in bytes.

**CPUID\_Fn00000005\_ECX [Monitor/MWait] (Core::X86::Cpuid::MonMWaitEcX)**

Read-only. Reset: Fixed,0000\_0003h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000005\_ECX

Bits	Description
31:2	Reserved.
1	<b>IBE</b> . Read-only. Reset: Fixed,1. Interrupt break-event.
0	<b>EMX</b> . Read-only. Reset: Fixed,1. Enumerate MONITOR/MWAIT extensions.

**CPUID\_Fn00000005\_EDX [Monitor/MWait] (Core::X86::Cpuid::MonMWaitEdx)**

Read-only. Reset: Fixed,0000\_0021h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000005\_EDX

Bits	Description
31:8	Reserved.
7:4	<b>MWaitC1SubStates</b> . Read-only. Reset: Fixed,2h. Number of C1 sub-cstates supported by MWAIT.
3:0	<b>MWaitC0SubStates</b> . Read-only. Reset: Fixed,1h. Number of C0 sub-cstates supported by MWAIT.

**CPUID\_Fn00000006\_EAX [Thermal and Power Management] (Core::X86::Cpuid::ThermalPwrMgmtEax)**

Read-only. Reset: Fixed,0000\_0004h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000006\_EAX

Bits	Description
31:3	Reserved.
2	<b>ARAT: always running APIC timer</b> . Read-only. Reset: Fixed,1. 1=Indicates support for APIC timer always running feature.
1:0	Reserved.

**CPUID\_Fn00000006\_EBX [Thermal and Power Management] (Core::X86::Cpuid::ThermalPwrMgmtEbx)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000006\_EBX

Bits	Description
31:0	Reserved.

**CPUID\_Fn00000006\_ECX [Thermal and Power Management] (Core::X86::Cpuid::ThermalPwrMgmtEcX)**

Read-only. Reset: Fixed,0000\_0001h.

These values can be over-written by Core::X86::Msrr::CPUID\_PWR\_THERM.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000006\_ECX

Bits	Description
31:1	Reserved.
0	<b>EffFreq: effective frequency interface</b> . Read-only. Reset: Fixed,1. 1=Indicates presence of Core::X86::Msrr::MPERF and Core::X86::Msrr::APERF.

**CPUID\_Fn00000006\_EDX [Thermal and Power Management] (Core::X86::Cpuid::ThermalPwrMgmtEdx)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000006\_EDX

Bits	Description
31:0	Reserved.

**CPUID\_Fn00000007\_EAX\_x00 [Structured Extended Feature Identifiers]  
(Core::X86::CpuId::StructExtFeatIdEax0)**

Read-only. Reset: Fixed, 0000\_0001h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000007\_EAX\_x00

Bits	Description
31:0	<b>StructExtFeatIdMax.</b> Read-only. Reset: Fixed, 0000_0001h. The largest CPUID Fn0000_0007 sub-function supported by the processor implementation.

**CPUID\_Fn00000007\_EBX\_x00 [Structured Extended Feature Identifiers]  
(Core::X86::CpuId::StructExtFeatIdEbx0)**

Read-only. Reset: Fixed, F1BF\_97ABh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000007\_EBX\_x00

Bits	Description
31	<b>AVX512VL.</b> Read-only. Reset: Fixed, 1. 1=Extends AVX512 instructions to 128 and 256 bits.
30	<b>AVX512BW.</b> Read-only. Reset: Fixed, 1. 1=AVX512BW packed integer instructions.
29	<b>SHA.</b> Read-only. Reset: Fixed, 1. 1=SHA Extensions available.
28	<b>AVX512CD.</b> Read-only. Reset: Fixed, 1. 1=AVX512 Conflict Detection for vectorizing loops supported.
27:25	Reserved.
24	<b>CLWB.</b> Read-only. Reset: Fixed, 1. 1=CLWB (Cache Line Write Back) instruction supported.
23	<b>CLFSHOPT.</b> Read-only. Reset: Fixed, 1. 1=CLFSHOPT (Optimized Cache Line Flush) instruction supported.
22	Reserved.
21	<b>AVX512_IFMA.</b> Read-only. Reset: Fixed, 1. 1=AVX512 integer fused mul-add instructions supported.
20	<b>SMAP.</b> Read-only. Reset: Fixed, 1. 1=Secure Mode Access Prevention is supported.
19	<b>ADX.</b> Read-only. Reset: Fixed, 1. 1=ADCX and ADOX instructions are supported.
18	<b>RDSEED.</b> Read-only. Reset: Fixed, 1. 1=RDSEED instruction is supported.
17	<b>AVX512DQ.</b> Read-only. Reset: Fixed, 1. 1=AVX512DQ packed integer instructions supported.
16	<b>AVX512F.</b> Read-only. Reset: Fixed, 1. 1=AVX512 Foundation supported.
15	<b>PQE.</b> Read-only. Reset: Fixed, 1. 1=Cache Allocation Technology is supported.
14:13	Reserved.
12	<b>PQM.</b> Read-only. Reset: Fixed, 1. 1=Platform QoS Monitoring is supported.
11	Reserved.
10	<b>INVPCID.</b> Read-only. Reset: Fixed, 1. 1=INVPCID instruction is supported.
9	<b>ERMS.</b> Read-only. Reset: Fixed, 1. 1=Enhanced REP MOVSB/STOSB is supported.
8	<b>BMI2.</b> Read-only. Reset: Fixed, 1. 1=Bit manipulation group 2 instructions are supported.
7	<b>SMEP.</b> Read-only. Reset: Fixed, 1. 1=Supervisor Mode Execution protection is supported.
6	Reserved.
5	<b>AVX2.</b> Read-only. Reset: Fixed, 1. 1=AVX extension support is supported.
4	Reserved.
3	<b>BMI1.</b> Read-only. Reset: Fixed, 1. 1=Bit manipulation group 1 instructions are supported.
2	Reserved.
1	<b>TSCADJUST.</b> Read-only. Reset: Fixed, 1. 1=Time-Stamp Counter Adjustment is supported.
0	<b>FSGSBASE.</b> Read-only. Reset: Fixed, 1. 1=FS and GS base read write instruction are supported.



**CPUID\_Fn00000007\_ECX\_x00 [Structured Extended Feature Identifier]  
(Core::X86::Cpuid::StructExtFeatIdEc0)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000007\_ECX\_x00

Bits	Description
31:29	Reserved.
28	<b>MOVDIR64B</b> . Read-only. Reset: Fixed,1. 1=MOVDIR64B instruction is supported.
27	<b>MOVDIRI</b> . Read-only. Reset: Fixed,1. 1=MOVDIRI instruction is supported.
26:25	Reserved.
24	<b>BusLock_Trap</b> . Read-only. Reset: Fixed,1. 1=Buslock Trap (generating a debug trap on bus locks that happen in CPL > 0) is supported when Core::X86::Msrr::DBG_CTL_MSR[BusLockTrapEn] is set.
23	Reserved.
22	<b>RDPID</b> . Read-only. Reset: Fixed,1. 1=Read Processor ID instruction is supported.
21:15	Reserved.
14	<b>AVX512_VPOPCNTDQ</b> . Read-only. Reset: Fixed,1. 1=AVX-512 VPOPCNTD/Q instructions supported.
13	Reserved.
12	<b>AVX512_BITALG</b> . Read-only. Reset: Fixed,1. 1=AVX-512 bit algorithm instructions VPSHUFBITQMB and VPOPCNTB/W supported.
11	<b>AVX512_VNNI</b> . Read-only. Reset: Fixed,1. 1=AVX512 vector neural network instructions supported.
10	<b>VPCLMULQDQ</b> . Read-only. Reset: X. 1=Vector VPCLMULQDQ instruction supported.
9	<b>VAES</b> . Read-only. Reset: X. 1=Vector VAES(ENC DEC), VAES(ENC DEC)LAST instruction supported.
8	<b>GFNI</b> . Read-only. Reset: Fixed,1. 1=Galois Field New Instructions supported.
7	<b>CET_SS</b> . Read-only. Reset: 1. 1=Shadow stack supported.
6	<b>AVX512_VBMI2</b> . Read-only. Reset: Fixed,1. 1=AVX512 vector byte permutation instruction 2 supported.
5:4	Reserved.
3	<b>PKU</b> . Read-only. Reset: Fixed,1. 1=Protection Keys are supported.
2	<b>UMIP</b> . Read-only. Reset: Fixed,1. 1=User Mode Instruction Prevention is supported.
1	<b>AVX512_VBMI</b> . Read-only. Reset: Fixed,1. 1=AVX512 vector byte permutation instruction are supported.
0	Reserved.

**CPUID\_Fn00000007\_EDX\_x00 [Structured Extended Feature Identifiers]  
(Core::X86::Cpuid::StructExtFeatIdEdx0)**

Read-only. Reset: Fixed,1000\_0110h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000007\_EDX\_x00

Bits	Description
31:29	Reserved.
28	<b>L1D_FLUSH</b> . Read-only. Reset: Fixed,1. 1=L1D_FLUSH supported in FLUSH_CMD MSR is supported.
27:9	Reserved.
8	<b>AVX512_VP2INTERSECT</b> . Read-only. Reset: Fixed,1. 1=VP2INTERSECT instruction is supported.
7:5	Reserved.
4	<b>FSRM</b> . Read-only. Reset: Fixed,1. 1=Fast Short REP MOVSB is supported.
3:0	Reserved.



**CPUID\_Fn00000007\_EAX\_x01 [Structured Extended Feature Identifiers]  
(Core::X86::Cpuid::StructExtFeatIdEax1)**

Read-only. Reset: Fixed, 0000\_0030h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000007\_EAX\_x01

Bits	Description
31:6	Reserved.
5	<b>AVX512_BF16.</b> Read-only. Reset: Fixed, 1. 1=BFLOAT16 instructions are supported.
4	<b>AVXVNNI.</b> Read-only. Reset: Fixed, 1. 1=AVX vector neural network instructions are supported.
3:0	Reserved.

**CPUID\_Fn00000007\_EBX\_x01 [Structured Extended Feature Identifiers]  
(Core::X86::Cpuid::StructExtFeatIdEbx1)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000007\_EBX\_x01

Bits	Description
31:0	Reserved.

**CPUID\_Fn00000007\_ECX\_x01 [Structured Extended Feature Identifier]  
(Core::X86::Cpuid::StructExtFeatIdEcX1)**

Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000007\_ECX\_x01

Bits	Description
31:0	Reserved.

**CPUID\_Fn00000007\_EDX\_x01 [Structured Extended Feature Identifiers]  
(Core::X86::Cpuid::StructExtFeatIdEdx1)**

Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000007\_EDX\_x01

Bits	Description
31:0	Reserved.

**CPUID\_Fn0000000B\_EAX\_x00 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEax0)**

Read-only.

CPUID Fn0000\_000B\_E[D,C,B,A]X\_x[2:0] specifies the hierarchy of logical cores from the SMT level through the processor socket level.

Software determines the presence of CPUID Fn0000\_000B if (CPUID Fn0000\_000B\_EBX\_x0[31:0] != 0). Software reads CPUID Fn0000\_000B\_E[C,B,A]X for ascending values of ECX until (CPUID Fn0000\_000B\_EBX[LogProcAtThisLevel] == 0).

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_EAX\_x00

Bits	Description
31:5	Reserved.
4:0	<b>CoreMaskWidth.</b> Read-only. Number of bits to shift ExtendedApicId right to get unique topology ID of the next instance of the current level type. Reset: SMT ? 01h : 00h.

**CPUID\_Fn0000000B\_EBX\_x00 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEbx0)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_EBX\_x00

Bits	Description
31:16	Reserved.
15:0	<b>LogProcAtThisLevel.</b> Read-only. Number of threads in a core. Reset: SMT ? 2 : 0001h.

**CPUID\_Fn0000000B\_ECX\_x00 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEcx0)**

Read-only. Reset: Fixed,0000\_0100h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_ECX\_x00

Bits	Description										
31:16	Reserved.										
15:8	<b>LevelType.</b> Read-only. Reset: Fixed,01h. <b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Invalid</td></tr> <tr> <td>01h</td><td>Thread</td></tr> <tr> <td>02h</td><td>Processor</td></tr> <tr> <td>FFh-03h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Invalid	01h	Thread	02h	Processor	FFh-03h	Reserved.
Value	Description										
00h	Invalid										
01h	Thread										
02h	Processor										
FFh-03h	Reserved.										
7:0	<b>EcxFVal.</b> Read-only. Reset: Fixed,00h. ECX input value.										

**CPUID\_Fn0000000B\_EAX\_x01 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEax1)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_EAX\_x01

Bits	Description
31:5	Reserved.
4:0	<b>CoreMaskWidth.</b> Read-only. Reset: XXXXXb. ExtendedApicId right shift value.

**CPUID\_Fn0000000B\_EBX\_x01 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEbx1)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_EBX\_x01

Bits	Description
31:16	Reserved.
15:0	<b>LogProcAtThisLevel.</b> Read-only. Reset: XXXXh. Number of logical cores in processor socket.

**CPUID\_Fn0000000B\_ECX\_x01 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEcx1)**

Read-only. Reset: Fixed,0000\_0201h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_ECX\_x01

Bits	Description										
31:16	Reserved.										
15:8	<b>LevelType.</b> Read-only. Reset: Fixed,02h. <b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Invalid</td></tr> <tr> <td>01h</td><td>Thread</td></tr> <tr> <td>02h</td><td>Processor</td></tr> <tr> <td>FFh-03h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Invalid	01h	Thread	02h	Processor	FFh-03h	Reserved.
Value	Description										
00h	Invalid										
01h	Thread										
02h	Processor										
FFh-03h	Reserved.										
7:0	<b>EcxFVal.</b> Read-only. Reset: Fixed,01h. ECX input value.										

**CPUID\_Fn0000000B\_EAX\_x02 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEax2)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_EAX\_x02

Bits	Description
31:5	Reserved.
4:0	<b>CoreMaskWidth.</b> Read-only. Reset: Fixed,00h. Zero indicates no more levels.

**CPUID\_Fn0000000B\_EBX\_x02 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEbx2)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_EBX\_x02

Bits	Description
31:16	Reserved.
15:0	<b>LogProcAtThisLevel.</b> Read-only. Reset: 0000h. Zero indicates no more levels.

**CPUID\_Fn0000000B\_ECX\_x02 [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEcxc2)**

Read-only. Reset: Fixed,0000\_0002h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_ECX\_x02

Bits	Description										
31:16	Reserved.										
15:8	<b>LevelType.</b> Read-only. Reset: Fixed,00h. <b>ValidValues:</b> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>00h</td><td>Invalid</td></tr> <tr> <td>01h</td><td>Thread</td></tr> <tr> <td>02h</td><td>Processor</td></tr> <tr> <td>FFh-03h</td><td>Reserved.</td></tr> </tbody> </table>	Value	Description	00h	Invalid	01h	Thread	02h	Processor	FFh-03h	Reserved.
Value	Description										
00h	Invalid										
01h	Thread										
02h	Processor										
FFh-03h	Reserved.										
7:0	<b>EcxcVal.</b> Read-only. Reset: Fixed,02h. ECX input value.										

**CPUID\_Fn0000000B\_EDX [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEdx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000B\_EDX

Bits	Description
31:0	<b>ExtendedLocalApicId: extended APIC ID.</b> Read-only. Reset: XXXX_XXXXh. Extended APIC_ID.

**CPUID\_Fn0000000D\_EAX\_x00 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEax00)**

Read-only. Reset: Fixed,0000\_02E7h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x00

Bits	Description		
31:0	<b>XFeatureSupportedMask[31:0]</b> . Read-only. Reset: Fixed,0000_02E7h. Each set bit indicates the corresponding bit in register XCR0[31:0] is settable.		
	<b>ValidValues:</b>		
	Bit	Name	Description
	[0]	X87	X87 Support.
	[1]	SSE	128-bit SSE Support.
	[2]	AVX	256-bit AVX support.
	[4:3]		Reserved.
	[5]	KREGS	KREGS
	[6]	ZMMHI	ZMMHI
	[7]	HIZMM	HIZMM
	[8]		Reserved.
	[9]	MPK	Memory Protection Keys. See Core::X86::Cpuid::StructExtFeatIdEcxc0[PKU] for the availability of MPK feature support.
[31:10]		Reserved.	

**CPUID\_Fn0000000D\_EBX\_x00 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEbx00)**

Read-only, Volatile.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x00

Bits	Description
31:0	<b>XFeatureEnabledSizeMax.</b> Read-only, Volatile. Reset: XXXX_XXXXh. <b>Description:</b> Size in bytes of an uncompact XSAVE/XRSTOR area for all features enabled in the XCR0 register. IF (XCR0[MPK] == 1) return EBX=0000_0988h ELSIF (XCR0[HIZMM] == 1) return EBX=0000_0980h ELSIF (XCR0[ZMMHI] == 1) return EBX=0000_0580h ELSIF (XCR0[KREGS] == 1) return EBX=0000_0380h ELSIF (XCR0[AVX] == 1) return EBX=0000_0340h ELSIF (XCR0[SSE] == 1) or (XCR0[X87] == 1) return EBX=0000_0240h // legacy header + X87/SSE size END

**CPUID\_Fn0000000D\_ECX\_x00 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEcX00)**

Read-only. Reset: Fixed, 0000\_0988h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x00

Bits	Description
31:0	<b>XFeatureSupportedSizeMax.</b> Read-only. Reset: Fixed, 0000_0988h. Size of legacy header + X87/SSE + AVX + KREGS + ZMMHI + HIZMM + MPK.

**CPUID\_Fn0000000D\_EDX\_x00 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEdx00)**

Read-only. Reset: Fixed, 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x00

Bits	Description
31:0	<b>XFeatureSupportedMask[63:32].</b> Read-only. Reset: Fixed, 0000_0000h. Each set bit indicates the corresponding bit in register XCR0[63:32] is settable.

**CPUID\_Fn0000000D\_EAX\_x01 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEax01)**

Read-only. Reset: Fixed, 0000\_000Fh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x01

Bits	Description
31:4	Reserved.
3	<b>XSAVES.</b> Read-only. Reset: Fixed, 1. XSAVES, XRSTORS, and XSS supported.
2	<b>XGETBV.</b> Read-only. Reset: Fixed, 1. XGETBV with ECX=1 supported.
1	<b>XSAVEC.</b> Read-only. Reset: Fixed, 1. XSAVEC and compact XRSTOR supported.
0	<b>XSAVEOPT.</b> Read-only. Reset: Fixed, 1. XSAVEOPT is available.

**CPUID\_Fn0000000D\_EBX\_x01 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEbx01)**

Read-only, Volatile.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x01

Bits	Description
31:0	<b>XFeatureEnabledSizeMax.</b> Read-only, Volatile. Reset: XXXX_XXXXh. <b>Description:</b> Reset is EBX = 0000_0240h + ((XCR0[AVX] == 1) ? 0000_0100h : 0) + ((XCR0[KREGS] == 1) ? 0000_0040h : 0) + ((XCR0[ZMMHI] == 1) ? 0000_0200h : 0) + ((XCR0[HIZMM] == 1) ? 0000_0400h : 0) + ((XCR0[MPK] == 1) ? 0000_0008h : 0) + ((XSS[CET_U] == 1) ? 0000_0010h : 0) + ((XSS[CET_S] == 1) ? 0000_0018h : 0).

**CPUID\_Fn0000000D\_ECX\_x01 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEcX01)**

Read-only. Reset: Fixed, 0000\_1800h.

Each set bit indicates the corresponding bit in register XSS[31:0] is settable.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x01

Bits	Description										
31:0	<b>MaskXss.</b> Read-only. Reset: Fixed, 0000_1800h. Mask[31:0] of settable XSS bits. <b>ValidValues:</b>										
	<table> <tr> <th>Bit</th><th>Description</th></tr> <tr> <td>[10:0]</td><td>Reserved.</td></tr> <tr> <td>[11]</td><td>CET for user mode.</td></tr> <tr> <td>[12]</td><td>CET for supervisor mode.</td></tr> <tr> <td>[31:13]</td><td>Reserved.</td></tr> </table>	Bit	Description	[10:0]	Reserved.	[11]	CET for user mode.	[12]	CET for supervisor mode.	[31:13]	Reserved.
Bit	Description										
[10:0]	Reserved.										
[11]	CET for user mode.										
[12]	CET for supervisor mode.										
[31:13]	Reserved.										

**CPUID\_Fn0000000D\_EDX\_x01 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEdx01)**

Read-only. Reset: Fixed, 0000\_0000h.

Each set bit indicates the corresponding bit in register XSS[63:32] is settable.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x01

Bits	Description
31:0	<b>MaskXss.</b> Read-only. Reset: Fixed, 0000_0000h. Mask[63:32] of settable XSS bits.

**CPUID\_Fn0000000D\_EAX\_x02 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEax02)**

Read-only. Reset: Fixed, 0000\_0100h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x02

Bits	Description
31:0	<b>YmmHiSaveStateOffset.</b> Read-only. Reset: Fixed, 0000_0100h. YMM[31:16] save state byte size.

**CPUID\_Fn0000000D\_EBX\_x02 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEbx02)**

Read-only. Reset: Fixed, 0000\_0240h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x02

Bits	Description
31:0	<b>YmmHiSaveStateOffset.</b> Read-only. Reset: Fixed, 0000_0240h. YMM[31:16] save state uncompact byte offset.

**CPUID\_Fn0000000D\_ECX\_x02 [Processor Extended State Enumeration]**  
**(Core::X86::Cpuid::ProcExtStateEnumEcxE02)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x02

Bits	Description
31:2	Reserved.
1	<b>YmmHiAligned.</b> Read-only. Reset: Fixed,0. 0=YMM_hi state (YMM[31:16]) is not automatically aligned to a 64-byte boundary on compacted saves/restores. 1=YMM_hi state (YMM[31:16]) is automatically aligned to a 64-byte boundary on compacted saves/restores.
0	<b>XStateSupervisor.</b> Read-only. Reset: Fixed,0. 1=This xstate is Supervisor State.

**CPUID\_Fn0000000D\_EDX\_x02 [Processor Extended State Enumeration]**  
**(Core::X86::Cpuid::ProcExtStateEnumEdxE02)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x02

Bits	Description
31:0	Reserved.

**CPUID\_Fn0000000D\_EAX\_x05 [Processor Extended State Enumeration]**  
**(Core::X86::Cpuid::ProcExtStateEnumEaxE05)**

Read-only. Reset: Fixed,0000\_0040h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x05

Bits	Description
31:0	<b>KREGSSaveStateSize.</b> Read-only. Reset: Fixed,0000_0040h. KREGS save state byte size.

**CPUID\_Fn0000000D\_EBX\_x05 [Processor Extended State Enumeration]**  
**(Core::X86::Cpuid::ProcExtStateEnumEbxE05)**

Read-only. Reset: Fixed,0000\_0340h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x05

Bits	Description
31:0	<b>KREGSSaveStateOffset.</b> Read-only. Reset: Fixed,0000_0340h. KREGS save state uncompact byte offset.

**CPUID\_Fn0000000D\_ECX\_x05 [Processor Extended State Enumeration]**  
**(Core::X86::Cpuid::ProcExtStateEnumEcxE05)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x05

Bits	Description
31:2	Reserved.
1	<b>KREGSAligned.</b> Read-only. Reset: Fixed,0. 0=KREGS state is not automatically aligned to a 64-byte boundary on compacted saves/restores. 1=KREGS state is automatically aligned to a 64-byte boundary on compacted saves/restores.
0	<b>XStateSupervisor.</b> Read-only. Reset: Fixed,0. 1=This xstate is Supervisor State.

**CPUID\_Fn0000000D\_EDX\_x05 [Processor Extended State Enumeration]**  
**(Core::X86::Cpuid::ProcExtStateEnumEdxE05)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x05

Bits	Description
31:0	Reserved.

**CPUID\_Fn0000000D\_EAX\_x06 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEax06)**

Read-only. Reset: Fixed,0000\_0200h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x06

Bits	Description
31:0	<b>ZMMHISaveStateSize.</b> Read-only. Reset: Fixed,0000_0200h. ZMMHI save state byte size.

**CPUID\_Fn0000000D\_EBX\_x06 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEbx06)**

Read-only. Reset: Fixed,0000\_0380h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x06

Bits	Description
31:0	<b>ZMMHISaveStateOffset.</b> Read-only. Reset: Fixed,0000_0380h. ZMMHI save state uncompact byte offset.

**CPUID\_Fn0000000D\_ECX\_x06 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEcX06)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x06

Bits	Description
31:2	Reserved.
1	<b>ZMMHISAligned.</b> Read-only. Reset: Fixed,0. 0=ZMMHI state is not automatically aligned to a 64-byte boundary on compacted saves/restores. 1=ZMMHI is automatically aligned to a 64-byte boundary on compacted saves/restores.
0	<b>XStateSupervisor.</b> Read-only. Reset: Fixed,0. 1=This xstate is Supervisor State.

**CPUID\_Fn0000000D\_EDX\_x06 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEdx06)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x06

Bits	Description
31:0	Reserved.

**CPUID\_Fn0000000D\_EAX\_x07 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEax07)**

Read-only. Reset: Fixed,0000\_0400h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x07

Bits	Description
31:0	<b>HIZMMSaveStateSize.</b> Read-only. Reset: Fixed,0000_0400h. HIZMM save state byte size.

**CPUID\_Fn0000000D\_EBX\_x07 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEbx07)**

Read-only. Reset: Fixed,0000\_0580h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x07

Bits	Description
31:0	<b>HIZMMSaveStateOffset.</b> Read-only. Reset: Fixed,0000_0580h. HIZMM save state uncompact byte offset.

**CPUID\_Fn0000000D\_ECX\_x07 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEcX07)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x07

Bits	Description
31:2	Reserved.
1	<b>HIZMMAligned.</b> Read-only. Reset: Fixed,0. 0=HIZMM state is not automatically aligned to a 64-byte boundary on compacted saves/restores. 1=HIZMM state is automatically aligned to a 64-byte boundary on compacted saves/restores.
0	<b>XStateSupervisor.</b> Read-only. Reset: Fixed,0. 1=This xstate is Supervisor State.

**CPUID\_Fn0000000D\_EDX\_x07 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEdX07)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x07

Bits	Description
31:0	Reserved.

**CPUID\_Fn0000000D\_EAX\_x09 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEax09)**

Read-only. Reset: Fixed,0000\_0008h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x09

Bits	Description
31:0	<b>MpkSaveStateSize.</b> Read-only. Reset: Fixed,0000_0008h. MPK save state byte size.

**CPUID\_Fn0000000D\_EBX\_x09 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEbx09)**

Read-only. Reset: Fixed,0000\_0980h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x09

Bits	Description
31:0	<b>MpKSaveStateOffset.</b> Read-only. Reset: Fixed,0000_0980h. MPK save state uncompact byte offset.

**CPUID\_Fn0000000D\_ECX\_x09 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEcX09)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x09

Bits	Description
31:2	Reserved.
1	<b>XState64ByteAligned.</b> Read-only. Reset: Fixed,0. 1=This xstate will always be 64-byte aligned in compacted memops.
0	<b>XStateSupervisor.</b> Read-only. Reset: Fixed,0. 1=This xstate is Supervisor State.

**CPUID\_Fn0000000D\_EDX\_x09 [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEdX09)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x09

Bits	Description
31:0	Reserved.



**CPUID\_Fn0000000D\_EAX\_x0B [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEax0B)**

Read-only. Reset: Fixed,0000\_0010h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x0B

Bits	Description
------	-------------

31:0	<b>CetUserSize.</b> Read-only. Reset: Fixed,0000_0010h. : CET user size.
------	--

**CPUID\_Fn0000000D\_EBX\_x0B [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEbx0B)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x0B

Bits	Description
------	-------------

31:0	<b>CetUserOffset.</b> Read-only. Reset: Fixed,0000_0000h. CET user byte offset.
------	---

**CPUID\_Fn0000000D\_ECX\_x0B [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEcxB)**

Read-only. Reset: Fixed,0000\_0001h.

Processor Extended State Enumeration for CET\_U.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x0B

Bits	Description
------	-------------

31:2	Reserved.
------	-----------

1	<b>XState64ByteAligned.</b> Read-only. Reset: Fixed,0. 1=This xstate will always be 64-byte aligned in compacted memops.
---	--

0	<b>XStateSupervisor.</b> Read-only. Reset: Fixed,1. 1=This xstate is Supervisor State.
---	--

**CPUID\_Fn0000000D\_EDX\_x0B [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEdxB)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x0B

Bits	Description
------	-------------

31:0	Reserved.
------	-----------

**CPUID\_Fn0000000D\_EAX\_x0C [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEax0C)**

Read-only. Reset: Fixed,0000\_0018h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EAX\_x0C

Bits	Description
------	-------------

31:0	<b>CetSprvrSize.</b> Read-only. Reset: Fixed,0000_0018h. CET supervisor size.
------	---

**CPUID\_Fn0000000D\_EBX\_x0C [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEbx0C)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EBX\_x0C

Bits	Description
------	-------------

31:0	<b>CetSprvrOffset.</b> Read-only. Reset: Fixed,0000_0000h. CET supervisor byte offset.
------	--

**CPUID\_Fn0000000D\_ECX\_x0C [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEcxC)**

Read-only. Reset: Fixed,0000\_0001h.

Processor Extended State Enumeration for CET\_S.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_ECX\_x0C

Bits	Description
31:2	Reserved.
1	<b>XState64ByteAligned</b> . Read-only. Reset: Fixed,0. 1=This xstate will always be 64-byte aligned in compacted memops.
0	<b>XStateSupervisor</b> . Read-only. Reset: Fixed,1. 1=This xstate is Supervisor State.

**CPUID\_Fn0000000D\_EDX\_x0C [Processor Extended State Enumeration]  
(Core::X86::Cpuid::ProcExtStateEnumEdxC)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000D\_EDX\_x0C

Bits	Description
31:0	Reserved.

**CPUID\_Fn0000000F\_EAX\_x00 [Resource Director Technology Monitor Capability]  
(Core::X86::Cpuid::RsrcDirTechMonCapEax0)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000F\_EAX\_x00

Bits	Description
31:0	Reserved.

**CPUID\_Fn0000000F\_EBX\_x00 [Resource Director Technology Monitor Capability]  
(Core::X86::Cpuid::RsrcDirTechMonCapEbx0)**

Read-only. Reset: Fixed,0000\_0FFFh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000F\_EBX\_x00

Bits	Description
31:0	<b>RmidMaxRange</b> . Read-only. Reset: Fixed,0000_0FFFh. RMID maximum within this processor for all types.

**CPUID\_Fn0000000F\_ECX\_x00 [Resource Director Technology Monitor Capability]  
(Core::X86::Cpuid::RsrcDirTechMonCapEcxC)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000F\_ECX\_x00

Bits	Description
31:0	Reserved.

**CPUID\_Fn0000000F\_EDX\_x00 [Resource Director Technology Monitor Capability]  
(Core::X86::Cpuid::RsrcDirTechMonCapEdxC)**

Read-only. Reset: Fixed,0000\_0002h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000F\_EDX\_x00

Bits	Description
31:2	Reserved.
1	<b>L3CacheRDT</b> . Read-only. Reset: Fixed,1. L3 Cache RDT Monitoring.
0	Reserved.

**CPUID\_Fn0000000F\_EAX\_x01 [Resource Director Technology L3 Monitor Capability]  
(Core::X86::Cpuid::RsrcDirTechMonCapEax1)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000F\_EAX\_x01

Bits	Description
31:9	Reserved.
8	<b>OverflowBit.</b> Read-only. Reset: Fixed,0. 1=Indicates Core::X86::Msr::QM_CTR bit[61] is an overflow bit.
7:0	<b>CounterSize.</b> Read-only. Reset: Fixed,20. Encode counter width offset from bit 24.
<b>ValidValues:</b>	
<b>Value</b>	<b>Description</b>
00h	Family/Model/Stepping should be used to determine counter size.
26h-01h	<Value>+24-bit counters.
FFh-27h	Reserved.

**CPUID\_Fn0000000F\_EBX\_x01 [Resource Director Technology L3 Monitor Capability]  
(Core::X86::Cpuid::RsrcDirTechMonCapEbx1)**

Read-only. Reset: Fixed,0000\_0040h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000F\_EBX\_x01

Bits	Description
31:0	<b>ConverFactor.</b> Read-only. Reset: Fixed,0000_0040h. Conversion Factor.

**CPUID\_Fn0000000F\_ECX\_x01 [Resource Director Technology L3 Monitor Capability]  
(Core::X86::Cpuid::RsrcDirTechMonCapEcxc1)**

Read-only. Reset: Fixed,0000\_0FFFh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000F\_ECX\_x01

Bits	Description
31:0	<b>RmidMaxRange.</b> Read-only. Reset: Fixed,0000_0FFFh. RMID Maximum Range of this resource.

**CPUID\_Fn0000000F\_EDX\_x01 [Resource Director Technology L3 Monitor Capability]  
(Core::X86::Cpuid::RsrcDirTechMonCapEdxc1)**

Read-only. Reset: Fixed,0000\_0007h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn0000000F\_EDX\_x01

Bits	Description
31:3	Reserved.
2	<b>L3CacheLocalBndwdthMon.</b> Read-only. Reset: Fixed,1. L3 Local Bandwidth monitoring.
1	<b>L3CacheTotalBndwdthMon.</b> Read-only. Reset: Fixed,1. L3 Total Bandwidth monitoring.
0	<b>L3CacheOccpncyMon.</b> Read-only. Reset: Fixed,1. L3 occupancy monitoring.

**CPUID\_Fn00000010\_EAX\_x00 [Resource Director Technology Allocation Enumeration]  
(Core::X86::Cpuid::RsrcDirTechAllocEnumEax0)**

Read-only. Reset: Fixed,0000\_0000h.

Software determines the presence of CPUID Fn0000\_0010 if (CPUID Fn0000\_0010\_EBX\_x0[31:0] != 0). Software reads CPUID Fn0000\_0010\_E[D,C,B,A]X for ascending values of ECX until (CPUID Fn0000\_0010\_EBX[LogProcAtThisLevel] == 0).

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000010\_EAX\_x00

Bits	Description
31:0	Reserved.

**CPUID\_Fn00000010\_EBX\_x00 [Resource Director Technology Allocation Enumeration]  
(Core::X86::Cpuid::RsrcDirTechAllocEnumEbx0)**

Read-only. Reset: 0000\_0002h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000010\_EBX\_x00

Bits	Description
31:3	Reserved.
2	<b>L2CacheAllocTech.</b> Read-only. Reset: 0. L2 Cache Allocation Technology.
1	<b>L3CacheAllocTech.</b> Read-only. Reset: 1. L3 Cache Allocation Technology.
0	Reserved.

**CPUID\_Fn00000010\_ECX\_x00 [Resource Director Technology Allocation Enumeration]  
(Core::X86::Cpuid::RsrcDirTechAllocEnumEcx0)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000010\_ECX\_x00

Bits	Description
31:0	Reserved.

**CPUID\_Fn00000010\_EDX\_x00 [Resource Director Technology Allocation Enumeration]  
(Core::X86::Cpuid::RsrcDirTechAllocEnumEdx0)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000010\_EDX\_x00

Bits	Description
31:0	Reserved.

**CPUID\_Fn00000010\_EAX\_x01 [Resource Director Technology L3 Allocation Enumeration]  
(Core::X86::Cpuid::RsrcDirTechAllocEnumEax1)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000010\_EAX\_x01

Bits	Description
31:5	Reserved.
4:0	<b>CapacityMask.</b> Read-only. Reset: Fixed,0Fh. Capacity bitmask length.

**CPUID\_Fn00000010\_EBX\_x01 [Resource Director Technology L3 Allocation Enumeration]  
(Core::X86::Cpuid::RsrcDirTechAllocEnumEbx1)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000010\_EBX\_x01

Bits	Description
31:0	<b>AllocUnits.</b> Read-only. Reset: 0000_0000h. Allocation Units.

**CPUID\_Fn00000010\_ECX\_x01 [Resource Director Technology L3 Allocation Enumeration]  
(Core::X86::Cpuid::RsrcDirTechAllocEnumEcx1)**

Read-only. Reset: Fixed,0000\_0004h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000010\_ECX\_x01

Bits	Description
31:3	Reserved.
2	<b>CDP.</b> Read-only. Reset: Fixed,1. Code and data prioritization.
1:0	Reserved.

**CPUID\_Fn00000010\_EDX\_x01 [Resource Director Technology L3 Allocation Enumeration] (Core::X86::Cpuid::RsrcDirTechAllocEnumEdx1)**

Read-only. Reset: Fixed,0000\_000Fh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn00000010\_EDX\_x01

Bits	Description
31:16	Reserved.
15:0	<b>HCS.</b> Read-only. Reset: Fixed,000Fh. Highest Class Of Service (COS) supported.

**CPUID\_Fn80000000\_EAX [Largest Extended Function Number] (Core::X86::Cpuid::LargExtFuncNum)**

Read-only. Reset: Fixed,8000\_0028h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000000\_EAX

Bits	Description
31:0	<b>LFuncExt: largest extended function.</b> Read-only. Reset: Fixed,8000_0028h. The largest CPUID extended function input value supported by the processor implementation.

**CPUID\_Fn80000000\_EBX [Processor Vendor (ASCII Bytes [3:0])] (Core::X86::Cpuid::ProcVendExtEbx)**

Read-only. Reset: Fixed,6874\_7541h.

Core::X86::Cpuid::ProcVendEbx and Core::X86::Cpuid::ProcVendExtEbx return the same value.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000000\_EBX

Bits	Description
31:0	<b>Vendor.</b> Read-only. Reset: Fixed,6874_7541h. ASCII Bytes [3:0] ("h t u A") of the string "AuthenticAMD".

**CPUID\_Fn80000000\_ECX [Processor Vendor (ASCII Bytes [11:8])] (Core::X86::Cpuid::ProcVendExtEcx)**

Read-only. Reset: Fixed,444D\_4163h.

Core::X86::Cpuid::ProcVendEcx and Core::X86::Cpuid::ProcVendExtEcx return the same value.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000000\_ECX

Bits	Description
31:0	<b>Vendor.</b> Read-only. Reset: Fixed,444D_4163h. ASCII Bytes [11:8] ("D M A c") of the string "AuthenticAMD".

**CPUID\_Fn80000000\_EDX [Processor Vendor (ASCII Bytes [7:4])] (Core::X86::Cpuid::ProcVendExtEdx)**

Read-only. Reset: Fixed,6974\_6E65h.

Core::X86::Cpuid::ProcVendEdx and Core::X86::Cpuid::ProcVendExtEdx return the same value.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000000\_EDX

Bits	Description
31:0	<b>Vendor.</b> Read-only. Reset: Fixed,6974_6E65h. ASCII Bytes [7:4] ("i t n e") of the string "AuthenticAMD".

**CPUID\_Fn80000001\_EAX [Family, Model, Stepping Identifiers] (Core::X86::Cpuid::FamModStepExt)**

Read-only.

Core::X86::Cpuid::FamModStep and Core::X86::Cpuid::FamModStepExt return the same value. See Core::X86::Cpuid::FamModStep.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000001\_EAX

Bits	Description
31:28	Reserved.
27:20	<b>ExtFamily: extended family.</b> Read-only. Reset: 0Bh. See Core::X86::Cpuid::FamModStep description of Family.
19:16	<b>ExtModel: extended model.</b> Read-only. Reset: 4h. See Core::X86::Cpuid::FamModStep description of ExtModel.
15:12	Reserved.
11:8	<b>BaseFamily.</b> Read-only. Reset: Fh. See Core::X86::Cpuid::FamModStep description of Family.
7:4	<b>BaseModel.</b> Read-only. Reset: Xh. Model numbers vary with product.
3:0	<b>Stepping.</b> Read-only. Reset: Xh. Processor stepping (revision) for a specific model.

**CPUID\_Fn80000001\_EBX [BrandId Identifier] (Core::X86::Cpuid::BrandId)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000001\_EBX

Bits	Description								
31:28	<b>PkgType: package type.</b> Read-only. Reset: Xh. Specifies the package type.								
	<b>ValidValues:</b>								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>3h-0h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>SP5</td></tr> <tr> <td>Fh-5h</td><td>Reserved.</td></tr> </table>	Value	Description	3h-0h	Reserved.	4h	SP5	Fh-5h	Reserved.
Value	Description								
3h-0h	Reserved.								
4h	SP5								
Fh-5h	Reserved.								
27:0	Reserved.								

CPUID_Fn80000001_ECX [Feature Identifiers] (Core::X86::Cpuid::FeatureExtIdEcX)	
Read-only.	
These values can be over-written by Core::X86::Msr::CPUID_ExtFeatures.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; CPUID_Fn80000001_ECX	
Bits	Description
31	Reserved.
30	<b>AdMskExtn: address mask extension support for instruction breakpoint.</b> Read-only. Reset: Fixed,1. Indicates support for address mask extension (to 32 bits and to all 4 DRs) for instruction breakpoints.
29	<b>MwaitExtended.</b> Read-only. Reset: !Core::X86::Msr::HWCR[MonMwaitDis]. 1=MWAITX and MONITORX capability is supported.
28	<b>PerfCtrExtLLC: Last Level Cache performance counter extensions.</b> Read-only. Reset: Fixed,1. 1=Indicates support for L3 performance counter extensions.
27	<b>PerfTsc.</b> Read-only. Reset: Fixed,0. Performance time-stamp counter supported.
26	<b>DataBreakpointExtension.</b> Read-only. Reset: Fixed,1. 1=Indicates data breakpoint support for Core::X86::Msr::DR0_ADDR_MASK, Core::X86::Msr::DR1_ADDR_MASK, Core::X86::Msr::DR2_ADDR_MASK and Core::X86::Msr::DR3_ADDR_MASK.
25	Reserved.
24	<b>PerfCtrExtDF: data fabric performance counter extensions support.</b> Read-only. Reset: Fixed,1. 1=Indicates support for Core::X86::Msr::DF_PERF_CTL and Core::X86::Msr::DF_PERF_CTR.
23	<b>PerfCtrExtCore: core performance counter extensions support.</b> Read-only. Reset: Fixed,1. 1=Indicates support for Core::X86::Msr::PERF_CTL0 - 5 and Core::X86::Msr::PERF_CTR.
22	<b>TopologyExtensions: topology extensions support.</b> Read-only. Reset: Fixed,1. 1=Indicates support for Core::X86::Cpuid::CachePropEax0 and Core::X86::Cpuid::ExtApicId.
21:18	Reserved.
17	<b>TCE.</b> Read-only. Reset: Fixed,1. Translation cache extension.
16	<b>FMA4.</b> Read-only. Reset: Fixed,0. Four-operand FMA instruction support.
15	<b>LWP.</b> Read-only. Reset: Fixed,0. Lightweight profiling support.
14	Reserved.
13	<b>WDT.</b> Read-only. Reset: Fixed,1. Watchdog timer support.
12	<b>SKINIT.</b> Read-only. Reset: Fixed,1. SKINIT and STGI support.
11	<b>XOP.</b> Read-only. Reset: Fixed,0. Extended operation support.
10	<b>IBS.</b> Read-only. Reset: Fixed,1. Instruction Based Sampling.
9	<b>OSVW.</b> Read-only. Reset: Fixed,1. OS Visible Work-around support.
8	<b>ThreeDNowPrefetch.</b> Read-only. Reset: Fixed,1. Prefetch and PrefetchW instructions.
7	<b>MisAlignSse.</b> Read-only. Reset: Fixed,1. Misaligned SSE Mode.
6	<b>SSE4A.</b> Read-only. Reset: Fixed,1. EXTRQ, INSERTQ, MOVNTSS, and MOVNTSD instruction support.
5	<b>ABM: advanced bit manipulation.</b> Read-only. Reset: Fixed,1. LZCNT instruction support.
4	<b>AltMovCr8.</b> Read-only. Reset: Fixed,1. LOCK MOV CR0 means MOV CR8.
3	<b>ExtApicSpace.</b> Read-only. Reset: Fixed,1. Extended APIC register space.
2	<b>SVM: Secure Virtual Mode feature.</b> Read-only. Reset: Fixed,1. Indicates support for: VMRUN, VMLOAD, VMSAVE, CLGI, VMMCALL, and INVLPGA.
1	<b>CmpLegacy.</b> Read-only. Reset: Fixed,(Core::X86::Cpuid::SizeId[NC] > 0). 0=Single core product (Core::X86::Cpuid::SizeId[NC] == 0). 1=Multi core product (Core::X86::Cpuid::SizeId[NC] != 0). Core multi-processing legacy mode.
0	<b>LahfSahf.</b> Read-only. Reset: Fixed,1. LAHF and SAHF instruction support in 64-bit mode.

**CPUID\_Fn80000001\_EDX [Feature Identifiers] (Core::X86::Cpuid::FeatureExtIdEdx)**

Read-only.

These values can be over-written by Core::X86::Msr::CPUID\_ExtFeatures.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000001\_EDX

Bits	Description
31	<b>ThreeDNow</b> . Read-only. Reset: Fixed,0. 3DNow! instructions.
30	<b>ThreeDNowExt</b> . Read-only. Reset: Fixed,0. AMD extensions to 3DNow! instructions.
29	<b>LM</b> . Read-only. Reset: Fixed,1. Long Mode.
28	Reserved.
27	<b>RDTSCP</b> . Read-only. Reset: Fixed,1. RDTSCP instruction.
26	<b>Page1GB</b> . Read-only. Reset: Fixed,1. 1-GB large page support.
25	<b>FFXSR</b> . Read-only. Reset: Fixed,1. FXSAVE and FXRSTOR instruction optimizations.
24	<b>FXSR</b> . Read-only. Reset: Fixed,1. FXSAVE and FXRSTOR instructions.
23	<b>MMX</b> . Read-only. Reset: Fixed,1. MMX instructions.
22	<b>MmxExt</b> . Read-only. Reset: Fixed,1. AMD extensions to MMX instructions.
21	Reserved.
20	<b>NX</b> . Read-only. Reset: Fixed,1. No-execute page protection.
19:18	Reserved.
17	<b>PSE36</b> . Read-only. Reset: Fixed,1. Page-size extensions.
16	<b>PAT</b> . Read-only. Reset: Fixed,1. Page attribute table.
15	<b>CMOV</b> . Read-only. Reset: Fixed,1. Conditional move instructions, CMOV, FCOMI, FCMOV.
14	<b>MCA</b> . Read-only. Reset: Fixed,1. Machine check architecture, MCG_CAP.
13	<b>PGE</b> . Read-only. Reset: Fixed,1. Page global extension, CR4.PGE.
12	<b>MTRR</b> . Read-only. Reset: Fixed,1. Memory-type range registers.
11	<b>SysCallSysRet</b> . Read-only. Reset: Fixed,1. SYSCALL and SYSRET instructions.
10	Reserved.
9	<b>APIC: advanced programmable interrupt controller (APIC) exists and is enabled</b> . Read-only. Reset: X. Reset is Core::X86::Msr::APIC_BAR[ApicEn].
8	<b>CMPXCHG8B</b> . Read-only. Reset: Fixed,1. CMPXCHG8B instruction.
7	<b>MCE</b> . Read-only. Reset: Fixed,1. Machine Check Exception, CR4.MCE.
6	<b>PAE</b> . Read-only. Reset: Fixed,1. Physical-address extensions (PAE).
5	<b>MSR</b> . Read-only. Reset: Fixed,1. Model-specific registers (MSRs), with RDMSR and WRMSR instructions.
4	<b>TSC</b> . Read-only. Reset: Fixed,1. Time stamp counter, RDTSC/RDTSCP instructions, CR4.TSD.
3	<b>PSE</b> . Read-only. Reset: Fixed,1. Page-size extensions (4 MB pages).
2	<b>DE</b> . Read-only. Reset: Fixed,1. Debugging extensions, IO breakpoints, CR4.DE.
1	<b>VME</b> . Read-only. Reset: Fixed,1. Virtual-mode enhancements.
0	<b>FPU</b> . Read-only. Reset: Fixed,1. x87 floating point unit on-chip.



**CPUID\_Fn80000002\_EAX [Processor Name String Identifier (Bytes [3:0])]  
(Core::X86::Cpuid::ProcNameStr0Eax)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000002\_EAX

Bits	Description
31:24	<b>ProcNameByte3.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString3]. Processor name, byte3.
23:16	<b>ProcNameByte2.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString2]. Processor name, byte2.
15:8	<b>ProcNameByte1.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString1]. Processor name, byte1.
7:0	<b>ProcNameByte0.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString0]. Processor name, byte0.

**CPUID\_Fn80000002\_EBX [Processor Name String Identifier (Bytes [7:4])]  
(Core::X86::Cpuid::ProcNameStr0Ebx)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000002\_EBX

Bits	Description
31:24	<b>ProcNameByte7.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString7]. Processor name, byte 7.
23:16	<b>ProcNameByte6.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString6]. Processor name, byte 6.
15:8	<b>ProcNameByte5.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString5]. Processor name, byte 5.
7:0	<b>ProcNameByte4.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString4]. Processor name, byte 4.

**CPUID\_Fn80000002\_ECX [Processor Name String Identifier (Bytes [11:8])]  
(Core::X86::Cpuid::ProcNameStr0Ecx)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000002\_ECX

Bits	Description
31:24	<b>ProcNameByte11.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n1[CpuNameString3]. Processor name, byte 11.
23:16	<b>ProcNameByte10.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n1[CpuNameString2]. Processor name, byte 10.
15:8	<b>ProcNameByte9.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n1[CpuNameString1]. Processor name, byte 9.
7:0	<b>ProcNameByte8.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n1[CpuNameString0]. Processor name, byte 8.

**CPUID\_Fn80000002\_EDX [Processor Name String Identifier (Bytes [15:12])]  
(Core::X86::Cpuid::ProcNameStr0Edx)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000002\_EDX

Bits	Description
31:24	<b>ProcNameByte15.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n1[CpuNameString7]. Processor name, byte 15.
23:16	<b>ProcNameByte14.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n1[CpuNameString6]. Processor name, byte 14.
15:8	<b>ProcNameByte13.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n1[CpuNameString5]. Processor name, byte 13.
7:0	<b>ProcNameByte12.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n1[CpuNameString4]. Processor name, byte 12.

**CPUID\_Fn80000003\_EAX [Processor Name String Identifier (Bytes [19:16])]  
(Core::X86::Cpuid::ProcNameStr1Eax)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n2.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000003\_EAX

Bits	Description
31:24	<b>ProcNameByte19.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n2[CpuNameString3]. Processor name, byte 19.
23:16	<b>ProcNameByte18.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n2[CpuNameString2]. Processor name, byte 18.
15:8	<b>ProcNameByte17.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n2[CpuNameString1]. Processor name, byte 17.
7:0	<b>ProcNameByte16.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n2[CpuNameString0]. Processor name, byte 16.

**CPUID\_Fn80000003\_EBX [Processor Name String Identifier (Bytes [23:20])]  
(Core::X86::Cpuid::ProcNameStr1Ebx)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n2.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000003\_EBX

Bits	Description
31:24	<b>ProcNameByte23.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n2[CpuNameString7]. Processor name, byte 23.
23:16	<b>ProcNameByte22.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n2[CpuNameString6]. Processor name, byte 22.
15:8	<b>ProcNameByte21.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n2[CpuNameString5]. Processor name, byte 21.
7:0	<b>ProcNameByte20.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n2[CpuNameString4]. Processor name, byte 20.

**CPUID\_Fn80000003\_ECX [Processor Name String Identifier (Bytes [27:24])]  
(Core::X86::Cpuid::ProcNameStr1EcX)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n3.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000003\_ECX

Bits	Description
31:24	<b>ProcNameByte27.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString3]. Processor name, byte 27.
23:16	<b>ProcNameByte26.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString2]. Processor name, byte 26.
15:8	<b>ProcNameByte25.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString1]. Processor name, byte 25.
7:0	<b>ProcNameByte24.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString0]. Processor name, byte 24.

**CPUID\_Fn80000003\_EDX [Processor Name String Identifier (Bytes [31:28])]  
(Core::X86::Cpuid::ProcNameStr1EdX)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n3.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000003\_EDX

Bits	Description
31:24	<b>ProcNameByte31.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString7]. Processor name, byte 31.
23:16	<b>ProcNameByte30.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString6]. Processor name, byte 30.
15:8	<b>ProcNameByte29.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString5]. Processor name, byte 29.
7:0	<b>ProcNameByte28.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString4]. Processor name, byte 28.

**CPUID\_Fn80000004\_EAX [Processor Name String Identifier (Bytes [35:32])]  
(Core::X86::Cpuid::ProcNameStr2Eax)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n4.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000004\_EAX

Bits	Description
31:24	<b>ProcNameByte35.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n4[CpuNameString3]. Processor name, byte 35.
23:16	<b>ProcNameByte34.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n4[CpuNameString2]. Processor name, byte 34.
15:8	<b>ProcNameByte33.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n4[CpuNameString1]. Processor name, byte 33.
7:0	<b>ProcNameByte32.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n4[CpuNameString0]. Processor name, byte 32.

**CPUID\_Fn80000004\_EBX [Processor Name String Identifier (Bytes [39:36])]  
(Core::X86::Cpuid::ProcNameStr2Ebx)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n4.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000004\_EBX

Bits	Description
31:24	<b>ProcNameByte39.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n4[CpuNameString7]. Processor name, byte 39.
23:16	<b>ProcNameByte38.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n4[CpuNameString6]. Processor name, byte 38.
15:8	<b>ProcNameByte37.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n4[CpuNameString5]. Processor name, byte 37.
7:0	<b>ProcNameByte36.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n4[CpuNameString4]. Processor name, byte 36.

**CPUID\_Fn80000004\_ECX [Processor Name String Identifier (Bytes [43:40])]  
(Core::X86::Cpuid::ProcNameStr2Ecx)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n5.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000004\_ECX

Bits	Description
31:24	<b>ProcNameByte43.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString3]. Processor name, byte 43.
23:16	<b>ProcNameByte42.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString2]. Processor name, byte 42.
15:8	<b>ProcNameByte41.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString1]. Processor name, byte 41.
7:0	<b>ProcNameByte40.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString0]. Processor name, byte 40.

**CPUID\_Fn80000004\_EDX [Processor Name String Identifier (Bytes [47:44])]  
(Core::X86::Cpuid::ProcNameStr2Edx)**

Read-only.

Is an alias of Core::X86::Msr::ProcNameString\_n5.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000004\_EDX

Bits	Description
31:24	<b>ProcNameByte47.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString7]. Processor name, byte 47.
23:16	<b>ProcNameByte46.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString6]. Processor name, byte 46.
15:8	<b>ProcNameByte45.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString5]. Processor name, byte 45.
7:0	<b>ProcNameByte44.</b> Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString4]. Processor name, byte 44.

**CPUID\_Fn80000005\_EAX [L1 TLB 2M/4M Identifiers] (Core::X86::Cpuid::L1Tlb2M4M)**

Read-only.

This function provides the processor's first level cache and TLB characteristics for each core.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000005\_EAX

Bits	Description
31:24	<b>L1DTlb2and4MAssoc: data TLB associativity for 2 MB and 4 MB pages.</b> Read-only. Reset: Fixed,FFh. See Core::X86::Cpuid::L1DcId[L1DcAssoc].
23:16	<b>L1DTlb2and4MSize: data TLB number of entries for 2 MB and 4 MB pages.</b> Read-only. Reset: Fixed,96. The value returned is for the number of entries available for the 2 MB page size; 4 MB pages require two 2 MB entries, so the number of entries available for the 4 MB page size is one-half the returned value.
15:8	<b>L1ITlb2and4MAssoc: instruction TLB associativity for 2 MB and 4 MB pages.</b> Read-only. Reset: Fixed,FFh. See Core::X86::Cpuid::L1DcId[L1DcAssoc].
7:0	<b>L1ITlb2and4MSize: instruction TLB number of entries for 2 MB and 4 MB pages.</b> Read-only. Reset: Fixed,64. The value returned is for the number of entries available for the 2 MB page size; 4 MB pages require two 2 MB entries, so the number of entries available for the 4 MB page size is one-half the returned value.

**CPUID\_Fn80000005\_EBX [L1 TLB 4K Identifiers] (Core::X86::Cpuid::L1Tlb4K)**

Read-only.

See Core::X86::Cpuid::L1Tlb2M4M.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000005\_EBX

Bits	Description
31:24	<b>L1DTlb4KAssoc.</b> Read-only. Reset: Fixed,FFh. Data TLB associativity for 4 KB pages. See Core::X86::Cpuid::L1DcId[L1DcAssoc].
23:16	<b>L1DTlb4KSize.</b> Read-only. Reset: Fixed,96. Data TLB number of entries for 4 KB pages.
15:8	<b>L1ITlb4KAssoc.</b> Read-only. Reset: Fixed,FFh. Instruction TLB associativity for 4 KB pages. See Core::X86::Cpuid::L1DcId[L1DcAssoc].
7:0	<b>L1ITlb4KSize.</b> Read-only. Reset: Fixed,64. Instruction TLB number of entries for 4 KB pages.

**CPUID\_Fn80000005\_ECX [L1 Data Cache Identifiers] (Core::X86::Cpuid::L1DcId)**

Read-only.

This function provides first level cache characteristics for each core.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000005\_ECX

Bits	Description														
31:24	<b>L1DcSize.</b> Read-only. Reset: Fixed,30h. L1 data cache size in KB.														
23:16	<b>L1DcAssoc.</b> Read-only. Reset: Fixed,0Ch. L1 data cache associativity. <b>ValidValues:</b>														
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>01h</td><td>1 way (direct mapped)</td></tr> <tr> <td>02h</td><td>2 way</td></tr> <tr> <td>03h</td><td>3 way</td></tr> <tr> <td>FEh-04h</td><td>&lt;Value&gt; way</td></tr> <tr> <td>FFh</td><td>Fully associative</td></tr> </table>	Value	Description	00h	Reserved.	01h	1 way (direct mapped)	02h	2 way	03h	3 way	FEh-04h	<Value> way	FFh	Fully associative
Value	Description														
00h	Reserved.														
01h	1 way (direct mapped)														
02h	2 way														
03h	3 way														
FEh-04h	<Value> way														
FFh	Fully associative														
15:8	<b>L1DcLinesPerTag.</b> Read-only. Reset: Fixed,01h. L1 data cache lines per tag.														
7:0	<b>L1DcLineSize.</b> Read-only. Reset: Fixed,64. L1 data cache line size in bytes.														

**CPUID\_Fn80000005\_EDX [L1 Instruction Cache Identifiers] (Core::X86::Cpuid::L1CId)**

Read-only.

This function provides first level cache characteristics for each core.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000005\_EDX

Bits	Description																
31:24	<b>L1IcSize.</b> Read-only. Reset: Fixed,32. L1 instruction cache size KB.																
23:16	<b>L1IcAssoc.</b> Read-only. Reset: Fixed,8. L1 instruction cache associativity.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>01h</td><td>1 way (direct mapped)</td></tr> <tr> <td>02h</td><td>2 way</td></tr> <tr> <td>03h</td><td>3 way</td></tr> <tr> <td>04h</td><td>4 way</td></tr> <tr> <td>FEh-05h</td><td>&lt;Value&gt; way</td></tr> <tr> <td>FFh</td><td>Fully associative</td></tr> </table>	Value	Description	00h	Reserved.	01h	1 way (direct mapped)	02h	2 way	03h	3 way	04h	4 way	FEh-05h	<Value> way	FFh	Fully associative
Value	Description																
00h	Reserved.																
01h	1 way (direct mapped)																
02h	2 way																
03h	3 way																
04h	4 way																
FEh-05h	<Value> way																
FFh	Fully associative																
15:8	<b>L1IcLinesPerTag.</b> Read-only. Reset: Fixed,01h. L1 instruction cache lines per tag.																
7:0	<b>L1IcLineSize.</b> Read-only. Reset: Fixed,64. L1 instruction cache line size in bytes.																

**CPUID\_Fn80000006\_EAX [L2 TLB 2M/4M Identifiers] (Core::X86::Cpuid::L2Tlb2M4M)**

Read-only.

This function provides the processor's second level cache and TLB characteristics for each core.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000006\_EAX

Bits	Description								
31:28	<b>L2DTlb2and4MAssoc: L2 data TLB associativity for 2 MB and 4 MB pages.</b> Read-only. Reset: Fixed,4h.								
	<b>ValidValues:</b>								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>3h-0h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>4 – 5 ways</td></tr> <tr> <td>Fh-5h</td><td>Reserved.</td></tr> </table>	Value	Description	3h-0h	Reserved.	4h	4 – 5 ways	Fh-5h	Reserved.
Value	Description								
3h-0h	Reserved.								
4h	4 – 5 ways								
Fh-5h	Reserved.								
27:16	<b>L2DTlb2and4MSize: L2 data TLB number of entries for 2 MB and 4 MB pages.</b> Read-only. Reset: Fixed,080h. The number of entries available for the 2 MB page size is 32 times the returned value; 4 MB pages require two 2 MB entries, so the number of entries available for the 4 MB page size is one-half the number of 2 MB entries.								
15:12	<b>L2ITlb2and4MAssoc: L2 instruction TLB associativity for 2 MB and 4 MB pages.</b> Read-only. Reset: Fixed,2.								
	<b>ValidValues:</b>								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>1h-0h</td><td>Reserved.</td></tr> <tr> <td>2h</td><td>2 ways</td></tr> <tr> <td>Fh-3h</td><td>Reserved.</td></tr> </table>	Value	Description	1h-0h	Reserved.	2h	2 ways	Fh-3h	Reserved.
Value	Description								
1h-0h	Reserved.								
2h	2 ways								
Fh-3h	Reserved.								
11:0	<b>L2ITlb2and4MSize: L2 instruction TLB number of entries for 2 MB and 4 MB pages.</b> Read-only. Reset: Fixed,040h. The number of entries available for the 2 MB page size is 32 times the returned value; 4 MB pages require two 2 MB entries, so the number of entries available for the 4 MB page size is one-half the number of 2 MB entries.								

**CPUID\_Fn80000006\_EBX [L2 TLB 4K Identifiers] (Core::X86::Cpuid::L2Tlb4K)**

Read-only.

This function provides the processor's second level cache and TLB characteristics for each core.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000006\_EBX

Bits	Description								
31:28	<b>L2DTlb4KAssoc.</b> Read-only. Reset: Fixed,6h. L2 data TLB associativity for 4 KB pages. <b>ValidValues:</b>								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>5h-0h</td><td>Reserved.</td></tr> <tr> <td>6h</td><td>8 – 15 ways</td></tr> <tr> <td>Fh-7h</td><td>Reserved.</td></tr> </table>	Value	Description	5h-0h	Reserved.	6h	8 – 15 ways	Fh-7h	Reserved.
Value	Description								
5h-0h	Reserved.								
6h	8 – 15 ways								
Fh-7h	Reserved.								
27:16	<b>L2DTlb4KSize.</b> Read-only. Reset: Fixed,128. The L2 data TLB number of entries for 4 KB pages is 32 times the returned value.								
15:12	<b>L2ITlb4KAssoc.</b> Read-only. Reset: Fixed,4. L2 instruction TLB associativity for 4 KB pages. <b>ValidValues:</b>								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>3h-0h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>4 – 5 ways</td></tr> <tr> <td>Fh-5h</td><td>Reserved.</td></tr> </table>	Value	Description	3h-0h	Reserved.	4h	4 – 5 ways	Fh-5h	Reserved.
Value	Description								
3h-0h	Reserved.								
4h	4 – 5 ways								
Fh-5h	Reserved.								
11:0	<b>L2ITlb4KSize.</b> Read-only. Reset: Fixed,64. The L2 instruction TLB number of entries for 4 KB pages is 32 times the returned value.								

**CPUID\_Fn80000006\_ECX [L2 Cache Identifiers] (Core::X86::Cpuid::L2CacheId)**

Read-only.

This function provides second level cache characteristics for each core.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000006\_ECX

Bits	Description																				
31:16	<b>L2Size.</b> Read-only. Reset: Fixed,0400h. L2 cache size in KB. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00FFh-0000h</td><td>Reserved.</td></tr> <tr> <td>0100h</td><td>256 KB</td></tr> <tr> <td>01FFh-0101h</td><td>Reserved.</td></tr> <tr> <td>0200h</td><td>512 KB</td></tr> <tr> <td>03FFh-0201h</td><td>Reserved.</td></tr> <tr> <td>0400h</td><td>1 MB</td></tr> <tr> <td>07FFh-0401h</td><td>Reserved.</td></tr> <tr> <td>0800h</td><td>2 MB</td></tr> <tr> <td>FFFFh-0801h</td><td>Reserved.</td></tr> </table>	Value	Description	00FFh-0000h	Reserved.	0100h	256 KB	01FFh-0101h	Reserved.	0200h	512 KB	03FFh-0201h	Reserved.	0400h	1 MB	07FFh-0401h	Reserved.	0800h	2 MB	FFFFh-0801h	Reserved.
Value	Description																				
00FFh-0000h	Reserved.																				
0100h	256 KB																				
01FFh-0101h	Reserved.																				
0200h	512 KB																				
03FFh-0201h	Reserved.																				
0400h	1 MB																				
07FFh-0401h	Reserved.																				
0800h	2 MB																				
FFFFh-0801h	Reserved.																				
15:12	<b>L2Assoc.</b> Read-only. Reset: Fixed,8. L2 cache associativity. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>7h-0h</td><td>Reserved.</td></tr> <tr> <td>8h</td><td>16 – 31 ways</td></tr> <tr> <td>Fh-9h</td><td>Reserved.</td></tr> </table>	Value	Description	7h-0h	Reserved.	8h	16 – 31 ways	Fh-9h	Reserved.												
Value	Description																				
7h-0h	Reserved.																				
8h	16 – 31 ways																				
Fh-9h	Reserved.																				
11:8	<b>L2LinesPerTag.</b> Read-only. Reset: Fixed,1h. L2 cache lines per tag.																				
7:0	<b>L2LineSize.</b> Read-only. Reset: Fixed,64. L2 cache line size in bytes.																				



**CPUID\_Fn80000006\_EDX [L3 Cache Identifiers] (Core::X86::Cpuid::L3CacheId)**

Read-only.

This function provides third level cache characteristics shared by all cores of a processor.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000006\_EDX

Bits	Description								
31:18	<b>L3Size: L3 cache size.</b> Read-only. Reset: XXXXh. The L3 cache size in 512 KB units. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0000h</td><td>Disabled.</td></tr> <tr> <td>3FFFh-0001h</td><td>(&lt;Value&gt; * 0.5) MB</td></tr> </table>	Value	Description	0000h	Disabled.	3FFFh-0001h	(<Value> * 0.5) MB		
Value	Description								
0000h	Disabled.								
3FFFh-0001h	(<Value> * 0.5) MB								
17:16	Reserved.								
15:12	<b>L3Assoc.</b> Read-only. Reset: Fixed,9h. There are insufficient available encodings to represent all possible L3 associativities. Please refer to Core::X86::Cpuid::CachePropEbx3[CacheNumWays]. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>8h-0h</td><td>Reserved.</td></tr> <tr> <td>9h</td><td>Invalid, not reported here.</td></tr> <tr> <td>Fh-Ah</td><td>Reserved.</td></tr> </table>	Value	Description	8h-0h	Reserved.	9h	Invalid, not reported here.	Fh-Ah	Reserved.
Value	Description								
8h-0h	Reserved.								
9h	Invalid, not reported here.								
Fh-Ah	Reserved.								
11:8	<b>L3LinesPerTag.</b> Read-only. Reset: Fixed,1h. L3 cache lines per tag.								
7:0	<b>L3LineSize.</b> Read-only. Reset: Fixed,64. L3 cache line size in bytes.								

**CPUID\_Fn80000007\_EAX [Reserved] (Core::X86::Cpuid::ProcFeedbackCap)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000007\_EAX

Bits	Description
31:0	Reserved.

**CPUID\_Fn80000007\_EBX [RAS Capabilities] (Core::X86::Cpuid::RasCap)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000007\_EBX

Bits	Description
31:4	Reserved.
3	<b>ScalableMca.</b> Read-only. Reset: Fixed,1. 0=Scalable MCA is not supported. 1=Scalable MCA is supported.
2	<b>HWA.</b> Read-only. Reset: Fixed,0. Hardware assert supported.
1	<b>SUCCOR: Software uncorrectable error containment and recovery capability.</b> Read-only. Reset: X. The processor supports software containment of uncorrectable errors through context synchronizing data poisoning and deferred error interrupts; MSR Core::X86::Msr::McaIntrCfg, MCA_STATUS[Deferred] and MCA_STATUS[Poison] exist.
0	<b>McaOverflowRecov: MCA overflow recovery support.</b> Read-only. Reset: Fixed,1. 0=MCA overflow conditions require software to shutdown the system. 1=MCA overflow conditions (MCA_STATUS[Overflow] == 1) are not fatal; software may safely ignore such conditions.

**CPUID\_Fn80000007\_ECX [Advanced Power Management Information] (Core::X86::Cpuid::ApmInfoEcX)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000007\_ECX

Bits	Description
31:0	<b>CpuPwrSampleTimeRatio.</b> Read-only. Reset: Fixed,0000_0000h. Specifies the ratio of the compute unit power accumulator sample period to the TSC counter period.

**CPUID\_Fn80000007\_EDX [Advanced Power Management Information] (Core::X86::Cpuid::ApmInfoEdx)**

Read-only.

This function provides advanced power management feature identifiers.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000007\_EDX

Bits	Description
31:16	Reserved.
15	<b>FastCPPC</b> . Read-only. Reset: Fixed,1. When set, indicates that Fast CPPC is supported.
14	<b>RAPL</b> . Read-only. Reset: Fixed,1. Running average power limit.
13	<b>ConnectedStandby</b> . Read-only. Reset: Fixed,1. Connected Standby.
12	<b>ProcPowerReporting</b> . Read-only. Reset: Fixed,0. Core power reporting interface supported.
11	<b>ProcFeedbackInterface: processor feedback interface</b> . Read-only. Reset: Fixed,0. 1=Indicates support for processor feedback interface; Core::X86::Cpuid::ProcFeedbackCap.
10	<b>EffFreqRO: read-only effective frequency interface</b> . Read-only. Reset: Fixed,1. Indicates presence of Core::X86::Msr::MPerfReadOnly and Core::X86::Msr::APerfReadOnly.
9	<b>CPB: core performance boost</b> . Read-only. Reset: X. 1=Indicates presence of Core::X86::Msr::HWCR[CpbDis] and support for core performance boost.
8	<b>TscInvariant: TSC invariant</b> . Read-only. Reset: Fixed,1. The TSC rate is invariant.
7	<b>HwPstate: hardware P-state control</b> . Read-only. Reset: Fixed,1. Core::X86::Msr::PStateCurLim, Core::X86::Msr::PStateCtl and Core::X86::Msr::PStateStat exist.
6	<b>OneHundredMHzSteps</b> . Read-only. Reset: Fixed,0. 100 MHz multiplier Control.
5	Reserved.
4	<b>TM</b> . Read-only. Reset: Fixed,1. Hardware thermal control (HTC)
3	<b>TTP</b> . Read-only. Reset: Fixed,1. THERMTRIP.
2:1	Reserved.
0	<b>TS</b> . Read-only. Reset: Fixed,1. Temperature sensor.

**CPUID\_Fn80000008\_EAX [Long Mode Address Size Identifiers] (Core::X86::Cpuid::LongModeInfo)**

Read-only. Reset: Fixed,0000\_3030h.

This provides information about the maximum physical and linear address width supported by the processor.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000008\_EAX

Bits	Description						
31:24	Reserved.						
23:16	<b>GuestPhysAddrSize</b> . Read-only. Reset: Fixed,00h. Maximum guest physical byte address size in bits.						
<b>ValidValues:</b>							
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>The maximum guest physical address size defined by PhysAddrSize.</td></tr> <tr> <td>FFh-01h</td><td>The maximum guest physical address size defined by GuestPhysAddrSize.</td></tr> </table>	Value	Description	00h	The maximum guest physical address size defined by PhysAddrSize.	FFh-01h	The maximum guest physical address size defined by GuestPhysAddrSize.
Value	Description						
00h	The maximum guest physical address size defined by PhysAddrSize.						
FFh-01h	The maximum guest physical address size defined by GuestPhysAddrSize.						
15:8	<b>LinAddrSize</b> . Read-only. Reset: Fixed,30h. Maximum linear byte address size in bits.						
7:0	<b>PhysAddrSize</b> . Read-only. Reset: Fixed,30h. Maximum physical byte address size in bits.						

**CPUID\_Fn80000008\_EBX [Extended Feature Extensions ID EBX] (Core::X86::Cpuid::FeatureExtIdEbx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000008\_EBX

Bits	Description
31	<b>BranchSample.</b> Read-only. Reset: Fixed,0. 1=Branch sampling feature supported. Branch sampling has been replaced with LBRv2.
30	<b>IBPB_RET.</b> Read-only. Reset: Fixed,1. When set, the processor clears the return address predictor on Core::X86::Msr::PRED_CMD[IBPB].
29	<b>BTC_NO.</b> Read-only. Reset: Fixed,1. Processor is not vulnerable to Branch Type Confusion.
28	<b>PSFD.</b> Read-only. Reset: Fixed,1. Predictive Store Forward Disable. See Core::X86::Msr::SPEC_CTRL[PSFD].
27	<b>CPPC.</b> Read-only. Reset: 1. Collaborative Processor Performance Control.
26:25	Reserved.
24	<b>SSBD: Speculative Store Bypass Disable.</b> Read-only. Reset: Fixed,1.
23	<b>PPIN: PPIN support.</b> Read-only. Reset: X. 0=PPIN capability is not supported; Core::X86::Msr::PPIN_CTL and Core::X86::Msr::PPIN are treated as RAZ. 1=Indicates that Protected Processor Inventory Number (PPIN) capability can be enabled for privileged system inventory agent to read PPIN from Core::X86::Msr::PPIN. Protected Processor Inventory Number support.
22:21	Reserved.
20	<b>EferLmsleUnsupported.</b> Read-only. Reset: Fixed,1. 1=Core::X86::Msr::EFER[LMSLE] is not supported, and MBZ.
19	<b>IbrsProvidesSameModeProtection.</b> Read-only. Reset: 1. IBRS provides Same Mode Protection.
18	<b>IbrsPreferred.</b> Read-only. Reset: 1. 1=IBRS is preferred over software solution.
17	<b>StibpAlwaysOn.</b> Read-only. Reset: 1. Single Thread Indirect Branch Prediction Mode has Enhanced Performance and may be left Always On.
16	Reserved.
15	<b>STIBP.</b> Read-only. Reset: 1. Single Thread Indirect Branch Prediction.
14	<b>IBRS.</b> Read-only. Reset: 1. Indirect Branch Restricted Speculation.
13	<b>INT_WBINVD.</b> Read-only. Reset: 1. Interruptible WBINVD,WBNOINVD.
12	<b>IBPB.</b> Read-only. Reset: 1. Indirect Branch Prediction Barrier.
11:10	Reserved.
9	<b>WBNOINVD.</b> Read-only. Reset: 1. WBNOINVD writes all modified cache lines in the internal caches of the processor back to memory leaving the line valid (clean) in the internal caches.
8	<b>MCOMMIT: memory commit.</b> Read-only. Reset: 0. Memory commit instruction support.
7	Reserved.
6	<b>MBE.</b> Read-only. Reset: Fixed,1. Memory Bandwidth Enforcement.
5	Reserved.
4	<b>RDPRU: read processor register at user level.</b> Read-only. Reset: Fixed,1. RDPRU instruction allows reading MPERF and APERF at user level.
3	<b>INVLPGb.</b> Read-only. Reset: Fixed,0. INVLPGb instruction broadcasts a TLB invalidate to all threads in the system.
2	<b>RstrFpErrPtrs.</b> Read-only. Reset: Fixed,1. 1=FXSAVE, XSAVE, FXSAVEOPT, XSAVEC, XSAVES always save error pointers and FXRSTOR, XRSTOR, XRSTORS always restore error pointers is supported.
1	<b>InstRetCntMsr: instructions retired count support.</b> Read-only. Reset: Fixed,1. 1=Core::X86::Msr::IRPerfCount supported.
0	<b>CLZERO: Clear Zero Instruction.</b> Read-only. Reset: Fixed,1. CLZERO instruction zero's out the 64 byte cache line specified in RAX. Note: CLZERO instruction operations are cache-line aligned and RAX[5:0] is ignored.

**CPUID\_Fn80000008\_ECX [Size Identifiers] (Core::X86::Cpuid::SizeId)**

Read-only.

This provides information about the number of threads supported by the processor.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000008\_ECX

Bits	Description
31:18	Reserved.
17:16	<b>PerfTscSize: performance time-stamp counter size.</b> Read-only. Reset: Fixed,0h.
15:12	<b>ApicIdSize: APIC ID size.</b> Read-only. Reset: Xh. The number of bits in the initial Core::X86::Apic::ApicId[ApicId] value that indicate thread ID within a package.
11:0	<b>NC: number of threads - 1.</b> Read-only. Reset: XXXh. The number of threads in the package is NC+1 (e.g., if NC=0, then there is one thread).

**CPUID\_Fn80000008\_EDX [Feature Extended Size Edx] (Core::X86::Cpuid::FeatureExtSizeEdx)**

Read-only. Reset: Fixed,0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000008\_EDX

Bits	Description
31:24	Reserved.
23:16	<b>RdpruMax.</b> Read-only. Reset: Fixed,01h. RDPRU Instruction max input supported.
15:0	Reserved.

**CPUID\_Fn8000000A\_EAX [SVM Revision and Feature Identification] (Core::X86::Cpuid::SvmRevFeatIdEax)**

Read-only. Reset: Fixed,0000\_0001h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000000A\_EAX

Bits	Description
31:8	Reserved.
7:0	<b>SvmRev.</b> Read-only. Reset: Fixed,01h. SVM revision.

**CPUID\_Fn8000000A\_EBX [SVM Revision and Feature Identification] (Core::X86::Cpuid::SvmRevFeatIdEbx)**

Read-only, Volatile. Reset: 0000\_8000h.

This provides SVM revision and feature information.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000000A\_EBX

Bits	Description
31:0	<b>NASID: number of address space identifiers (ASID).</b> Read-only, Volatile. Reset: 0000_8000h.

**CPUID\_Fn8000000A\_EDX [SVM Revision and Feature Identification] (Core::X86::Cpuid::SvmRevFeatIdEdx)**

Read-only.

This provides SVM feature information.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000000A\_EDX

Bits	Description
31	<b>EnhancedShutdownIntercept.</b> Read-only. Reset: Fixed,1. Support for EXITINFO1 on shutdown intercept and nested shutdown intercepts will result in a non-interceptible shutdown.
30	<b>IdleHltIntercept.</b> Read-only. Reset: Fixed,1. Indicates support for intercepting HLT conditionally if there is no virtual interrupt pending
29	<b>GuestBusLockThreshold.</b> Read-only. Reset: Fixed,1. Allows hypervisor to intercept a guest after it has executed a programmable number of bus locks.
28	<b>VmcbAddrChkChg.</b> Read-only. Reset: Fixed,1. VMCB address check change
27	<b>ExtLvtOffsetFaultChg.</b> Read-only. Reset: Fixed,1. 1=Read/Write fault behavior for the extended LVT offsets (APIC addresses 0x500-0x530) changed to Read Allowed, Write #MVEXIT (trap).
26	<b>IbsVirt.</b> Read-only. Reset: Fixed,1. IBS Virtualization
25	<b>NmiVirt.</b> Read-only. Reset: Fixed,1. Guest NMI Virtualization
24	Reserved.
23	<b>HOST_MCE_OVERRIDE.</b> Read-only. Reset: Fixed,1. 1=If hCR4:MCE == 1 and gCR4:MCE == 0, machine check exceptions (#MC) in guest do not cause shutdown and are always intercepted.
22	Reserved.
21	<b>AllowNonWriteAbleGPT.</b> Read-only. Reset: Fixed,1. 1=Indicates support for Non-Writeable Guest Pages for NPT.
20	<b>GuestSpecCtrl.</b> Read-only. Reset: Fixed,1. 1=Indicates support for Guest Spec_ctl.
19	<b>SupervisorShadowStack.</b> Read-only. Reset: Fixed,1.
18	<b>X2AVIC: virtualized X2APIC.</b> Read-only. Reset: 0. 1=Virtualized X2APIC is supported.
17	<b>GMET.</b> Read-only. Reset: Fixed,1. Guest Mode Execute Trap.
16	<b>vGIF.</b> Read-only. Reset: Fixed,1. Virtualized GIF.
15	<b>V_VMSAVE_VMLOAD.</b> Read-only. Reset: Fixed,1. Virtualized VMLOAD and VMSAVE.
14	Reserved.
13	<b>AVIC: AMD virtual interrupt controller.</b> Read-only. Reset: 0. 1=Support indicated for SVM mode virtualized interrupt controller; Indicates support for Core::X86::Msr::AvicDoorbell.
12	<b>PauseFilterThreshold.</b> Read-only. Reset: Fixed,1.
11	Reserved.
10	<b>PauseFilter.</b> Read-only. Reset: Fixed,1. Pause intercept filter.
9	Reserved.
8	<b>PerfCtrVirt.</b> Read-only. Reset: Fixed,1. Indicates support for virtualization of Performance Monitor Counters.
7	<b>DecodeAssists.</b> Read-only. Reset: Fixed,1.
6	<b>FlushByAsid.</b> Read-only. Reset: Fixed,1.
5	<b>VmcbClean.</b> Read-only. Reset: Fixed,1. VMCB clean bits.
4	<b>TscRateMsr: MSR based TSC rate control.</b> Read-only. Reset: Fixed,1. 1=Indicates support for TSC ratio Core::X86::Msr::TscRateMsr. MSR based TSC rate control.
3	<b>NRIPS.</b> Read-only. Reset: Fixed,1. NRIP Save.
2	<b>SVML.</b> Read-only. Reset: Fixed,1. SVM lock.
1	<b>LbrVirt.</b> Read-only. Reset: Fixed,1. LBR virtualization.
0	<b>NP.</b> Read-only. Reset: Fixed,1. Nested Paging.

**CPUID\_Fn80000019\_EAX [L1 TLB 1G Identifiers] (Core::X86::Cpuid::L1Tlb1G)**

Read-only.

This function provides first level TLB characteristics for 1GB pages.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000019\_EAX

Bits	Description
31:28	<b>L1DTlb1GAssoc</b> : L1 data TLB associativity for 1 GB pages. Read-only. Reset: Fixed,Fh. See Core::X86::Cpuid::L2CacheId[L2Assoc].
27:16	<b>L1DTlb1GSize</b> . Read-only. Reset: Fixed,96. L1 data TLB number of entries for 1 GB pages.
15:12	<b>L1ITlb1GAssoc</b> . Read-only. Reset: Fixed,Fh. L1 instruction TLB associativity for 1 GB pages. See Core::X86::Cpuid::L2CacheId[L2Assoc].
11:0	<b>L1ITlb1GSize</b> . Read-only. Reset: Fixed,64. L1 instruction TLB number of entries for 1 GB pages.

**CPUID\_Fn80000019\_EBX [L2 TLB 1G Identifiers] (Core::X86::Cpuid::L2Tlb1G)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000019\_EBX

Bits	Description
31:28	<b>L2DTlb1GAssoc</b> . Read-only. Reset: Fixed,4. L2 data TLB associativity for 1 GB pages.
27:16	<b>L2DTlb1GSize</b> . Read-only. Reset: Fixed,020h. L2 data TLB number of entries for 1 GB pages.
15:12	<b>L2ITlb1GAssoc</b> . Read-only. Reset: Fixed,0h. L2 instruction TLB associativity for 1 GB pages.
11:0	<b>L2ITlb1GSize</b> . Read-only. Reset: Fixed,000h. L2 instruction TLB number of entries for 1 GB pages.

**CPUID\_Fn8000001A\_EAX [Performance Optimization Identifiers] (Core::X86::Cpuid::PerfOptId)**

Read-only. Reset: Fixed,0000\_000Ah.

This function returns performance related information.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001A\_EAX

Bits	Description
31:4	Reserved.
3	<b>FP512</b> . Read-only. Reset: Fixed,1. When set, the internal FP/SIMD execution datapath is 512-bits wide.
2	<b>FP256</b> . Read-only. Reset: Fixed,0. When set, the internal FP/SIMD execution datapath is 256-bits wide.
1	<b>MOVU</b> . Read-only. Reset: Fixed,1. MOVU SSE instructions are more efficient and should be preferred to SSE MOVL/MOVH. MOVUPS is more efficient than MOVLPS/MOVHPS. MOVUPD is more efficient than MOVLPS/MOVHPS.
0	<b>FP128</b> . Read-only. Reset: Fixed,0. When set, the internal FP/SIMD execution datapath is 128-bits wide.

**CPUID\_Fn8000001B\_EAX [Instruction Based Sampling Identifiers] (Core::X86::Cpuid::IbsIdEax)**

Read-only. Reset: Fixed, 0008\_1BFFh.

This function returns IBS feature information.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001B\_EAX

Bits	Description
31:20	Reserved.
19	<b>IbsUpdtdDtlbStats.</b> Read-only. Reset: Fixed, 1. Simplified DTLB page size and miss reporting in IBS_OP_DATA3.
18:13	Reserved.
12	<b>IbsLoadLatencyFiltering.</b> Read-only. Reset: Fixed, 1. Indicates support for filtering of IBS execution samples based on load latency.
11	<b>Zen4IbsExtensions.</b> Read-only. Reset: Fixed, 1. Fetch and Op IBS support IBS extensions added with Zen4: Core::X86::Msr::IBS_FETCH_CTL[IbsFetchL3Miss], Core::X86::Msr::IBS_FETCH_CTL[IbsFetchOcMiss], Core::X86::Msr::IBS_FETCH_CTL[IbsL3MissOnly] and Core::X86::Msr::IBS_OP_DATA2 DataSrc extension.
10	<b>IbsOpData4.</b> Read-only. Reset: Fixed, 0. IBS op data 4 MSR supported.
9	<b>IbsFetchCtlExtd: IBS fetch control extended MSR supported.</b> Read-only. Reset: Fixed, 1. Indicates support for Core::X86::Msr::IC_IBS_EXTD_CTL.
8	<b>OpFuse: fused branch op indication supported.</b> Read-only. Reset: Fixed, 1. Indicates support for Core::X86::Msr::IBS_OP_DATA[IbsOpFuse].
7	<b>RipInvalidChk: invalid RIP indication supported.</b> Read-only. Reset: Fixed, 1. Indicates support for Core::X86::Msr::IBS_OP_DATA[IbsRipInvalid].
6	<b>OpCntExt: IbsOpCurCnt and IbsOpMaxCnt extend by 7 bits.</b> Read-only. Reset: Fixed, 1. Indicates support for Core::X86::Msr::IBS_OP_CTL[IbsOpCurCnt[26:20], IbsOpMaxCnt[26:20]].
5	<b>BrnTrgt.</b> Read-only. Reset: Fixed, 1. Branch target address reporting supported.
4	<b>OpCnt.</b> Read-only. Reset: Fixed, 1. Op counting mode supported.
3	<b>RdWrOpCnt.</b> Read-only. Reset: Fixed, 1. Read write of op counter supported.
2	<b>OpSam.</b> Read-only. Reset: Fixed, 1. IBS execution sampling supported.
1	<b>FetchSam.</b> Read-only. Reset: Fixed, 1. IBS fetch sampling supported.
0	<b>IBSFFV.</b> Read-only. Reset: Fixed, 1. IBS feature flags valid.



**CPUID\_Fn8000001D\_EAX\_x00 [Cache Properties (DC)] (Core::X86::Cpuid::CachePropEax0)**

Core::X86::Cpuid::CachePropEax0 reports topology information for the DC.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EAX\_x00

Bits	Description												
31:26	Reserved.												
25:14	<b>NumSharingCache: number of logical processors sharing cache.</b> Read-only. Reset: XXXh. The number of logical processors sharing this cache is NumSharingCache + 1.												
13:10	Reserved.												
9	<b>FullyAssociative: fully associative cache.</b> Read-only. Reset: Fixed,0. 1=Cache is fully associative.												
8	<b>SelfInitialization: cache is self-initializing.</b> Read-only. Reset: Fixed,1. 1=Cache is self initializing; cache does not need software initialization.												
7:5	<b>CacheLevel: cache level.</b> Read-only. Reset: Fixed,1h. Identifies the cache level. <b>ValidValues:</b>												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Reserved.</td></tr> <tr> <td>1h</td><td>Level 1</td></tr> <tr> <td>2h</td><td>Level 2</td></tr> <tr> <td>3h</td><td>Level 3</td></tr> <tr> <td>7h-4h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	Reserved.	1h	Level 1	2h	Level 2	3h	Level 3	7h-4h	Reserved.
Value	Description												
0h	Reserved.												
1h	Level 1												
2h	Level 2												
3h	Level 3												
7h-4h	Reserved.												
4:0	<b>CacheType: cache type.</b> Read-only. Reset: Fixed,01h. Identifies the type of cache. <b>ValidValues:</b>												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Null; no more caches.</td></tr> <tr> <td>01h</td><td>Data cache.</td></tr> <tr> <td>02h</td><td>Instruction cache.</td></tr> <tr> <td>03h</td><td>Unified cache.</td></tr> <tr> <td>1Fh-04h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Null; no more caches.	01h	Data cache.	02h	Instruction cache.	03h	Unified cache.	1Fh-04h	Reserved.
Value	Description												
00h	Null; no more caches.												
01h	Data cache.												
02h	Instruction cache.												
03h	Unified cache.												
1Fh-04h	Reserved.												

**CPUID\_Fn8000001D\_EAX\_x01 [Cache Properties (IC)] (Core::X86::Cpuid::CachePropEax1)**

Read-only.

Core::X86::Cpuid::CachePropEax1 reports topology information for the IC. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EAX\_x01

Bits	Description
31:26	Reserved.
25:14	<b>NumSharingCache: number of logical processors sharing cache.</b> Read-only. Reset: XXXh. See Core::X86::Cpuid::CachePropEax0[NumSharingCache].
13:10	Reserved.
9	<b>FullyAssociative: fully associative cache.</b> Read-only. Reset: Fixed,0. See Core::X86::Cpuid::CachePropEax0[FullyAssociative].
8	<b>SelfInitialization: cache is self-initializing.</b> Read-only. Reset: Fixed,1. See Core::X86::Cpuid::CachePropEax0[SelfInitialization].
7:5	<b>CacheLevel: cache level.</b> Read-only. Reset: Fixed,1h. Identifies the cache level. See Core::X86::Cpuid::CachePropEax0[CacheLevel].
4:0	<b>CacheType: cache type.</b> Read-only. Reset: Fixed,02h. See Core::X86::Cpuid::CachePropEax0[CacheType].



**CPUID\_Fn8000001D\_EAX\_x02 [Cache Properties (L2)] (Core::X86::Cpuid::CachePropEax2)**

Read-only.

Core::X86::Cpuid::CachePropEax2 reports topology information for the L2. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EAX\_x02

Bits	Description
31:26	Reserved.
25:14	<b>NumSharingCache: number of logical processors sharing cache.</b> Read-only. Reset: XXXh. Core::X86::Cpuid::CachePropEax0[NumSharingCache].
13:10	Reserved.
9	<b>FullyAssociative: fully associative cache.</b> Read-only. Reset: Fixed,0. Core::X86::Cpuid::CachePropEax0[FullyAssociative].
8	<b>SelfInitialization: cache is self-initializing.</b> Read-only. Reset: Fixed,1. Core::X86::Cpuid::CachePropEax0[SelfInitialization].
7:5	<b>CacheLevel: cache level.</b> Read-only. Reset: Fixed,2h. Identifies the cache level. Core::X86::Cpuid::CachePropEax0[CacheLevel].
4:0	<b>CacheType: cache type.</b> Read-only. Reset: Fixed,03h. Core::X86::Cpuid::CachePropEax0[CacheType].

**CPUID\_Fn8000001D\_EAX\_x03 [Cache Properties (L3)] (Core::X86::Cpuid::CachePropEax3)**

Read-only.

Core::X86::Cpuid::CachePropEax3 reports topology information for the L3.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EAX\_x03

Bits	Description
31:26	Reserved.
25:14	<b>NumSharingCache: number of logical processors sharing cache.</b> Read-only. Reset: XXXh. The number of logical processors sharing this cache is NumSharingCache + 1.
13:10	Reserved.
9	<b>FullyAssociative: fully associative cache.</b> Read-only. Reset: Fixed,0. Core::X86::Cpuid::CachePropEax0[FullyAssociative].
8	<b>SelfInitialization: cache is self-initializing.</b> Read-only. Reset: Fixed,1. Core::X86::Cpuid::CachePropEax0[SelfInitialization].
7:5	<b>CacheLevel: cache level.</b> Read-only. Reset: Fixed,3h. Identifies the cache level. Core::X86::Cpuid::CachePropEax0[CacheLevel].
4:0	<b>CacheType: cache type.</b> Read-only. Reset: Fixed,03h. Core::X86::Cpuid::CachePropEax0[CacheType].

**CPUID\_Fn8000001D\_EAX\_x04 [Cache Properties Null] (Core::X86::Cpuid::CachePropEax4)**

Read-only. Reset: Fixed,0000\_0000h.

Core::X86::Cpuid::CachePropEax4 reports done/null. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EAX\_x04

Bits	Description
31:5	Reserved.
4:0	<b>CacheType: cache type.</b> Read-only. Reset: Fixed,00h. Core::X86::Cpuid::CachePropEax0[CacheType].

**CPUID\_Fn8000001D\_EBX\_x00 [Cache Properties (DC)] (Core::X86::Cpuid::CachePropEbx0)**

Read-only. Reset: Fixed,02C0\_003Fh.

Core::X86::Cpuid::CachePropEbx0 reports topology information for the DC. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EBX\_x00

Bits	Description
31:22	<b>CacheNumWays: cache number of ways.</b> Read-only. Reset: Fixed,00Bh. Cache number of ways is CacheNumWays + 1.
21:12	<b>CachePhysPartitions: cache physical line partitions.</b> Read-only. Reset: Fixed,000h. Cache partitions is CachePhysPartitions + 1.
11:0	<b>CacheLineSize: cache line size in bytes.</b> Read-only. Reset: Fixed,03Fh. Cache line size in bytes is CacheLineSize + 1.

**CPUID\_Fn8000001D\_EBX\_x01 [Cache Properties (IC)] (Core::X86::Cpuid::CachePropEbx1)**

Read-only. Reset: Fixed,01C0\_003Fh.

Core::X86::Cpuid::CachePropEbx1 reports topology information for the IC. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EBX\_x01

Bits	Description
31:22	<b>CacheNumWays: cache number of ways.</b> Read-only. Reset: Fixed,007h. Core::X86::Cpuid::CachePropEbx0[CacheNumWays].
21:12	<b>CachePhysPartitions: cache physical line partitions.</b> Read-only. Reset: Fixed,000h. Core::X86::Cpuid::CachePropEbx0[CachePhysPartitions].
11:0	<b>CacheLineSize: cache line size in bytes.</b> Read-only. Reset: Fixed,03Fh. Core::X86::Cpuid::CachePropEbx0[CacheLineSize].

**CPUID\_Fn8000001D\_EBX\_x02 [Cache Properties (L2)] (Core::X86::Cpuid::CachePropEbx2)**

Read-only. Reset: Fixed,03C0\_003Fh.

Core::X86::Cpuid::CachePropEbx2 reports topology information for the L2. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EBX\_x02

Bits	Description
31:22	<b>CacheNumWays: cache number of ways.</b> Read-only. Reset: Fixed,00Fh. See Core::X86::Cpuid::CachePropEbx0[CacheNumWays].
21:12	<b>CachePhysPartitions: cache physical line partitions.</b> Read-only. Reset: Fixed,000h. See Core::X86::Cpuid::CachePropEbx0[CachePhysPartitions].
11:0	<b>CacheLineSize: cache line size in bytes.</b> Read-only. Reset: Fixed,03Fh. See Core::X86::Cpuid::CachePropEbx0[CacheLineSize].

**CPUID\_Fn8000001D\_EBX\_x03 [Cache Properties (L3)] (Core::X86::Cpuid::CachePropEbx3)**

Read-only. Reset: Fixed,03C0\_003Fh.

Core::X86::Cpuid::CachePropEbx3 reports topology information for the L3. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EBX\_x03

Bits	Description
31:22	<b>CacheNumWays: cache number of ways.</b> Read-only. Reset: Fixed,00Fh. See Core::X86::Cpuid::CachePropEbx0[CacheNumWays].
21:12	<b>CachePhysPartitions: cache physical line partitions.</b> Read-only. Reset: Fixed,000h. See Core::X86::Cpuid::CachePropEbx0[CachePhysPartitions].
11:0	<b>CacheLineSize: cache line size in bytes.</b> Read-only. Reset: Fixed,03Fh. See Core::X86::Cpuid::CachePropEbx0[CacheLineSize].

**CPUID\_Fn8000001D\_EBX\_x04 [Cache Properties Null] (Core::X86::Cpuid::CachePropEbx4)**

Read-only. Reset: Fixed,0000\_0000h.

Core::X86::Cpuid::CachePropEax4 reports done/null. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EBX\_x04

**Bits Description**

31:0 Reserved.

**CPUID\_Fn8000001D\_ECX\_x00 [Cache Properties (DC)] (Core::X86::Cpuid::CachePropEcxc0)**

Read-only. Reset: Fixed,0000\_003Fh.

Core::X86::Cpuid::CachePropEcxc0 reports topology information for the DC. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_ECX\_x00

**Bits Description**31:0 **CacheNumSets: cache number of sets.** Read-only. Reset: Fixed,0000\_003Fh. Cache number of sets is CacheNumSets + 1.**CPUID\_Fn8000001D\_ECX\_x01 [Cache Properties (IC)] (Core::X86::Cpuid::CachePropEcxc1)**

Read-only. Reset: Fixed,0000\_003Fh.

Core::X86::Cpuid::CachePropEcxc1 reports topology information for the IC. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_ECX\_x01

**Bits Description**31:0 **CacheNumSets: cache number of sets.** Read-only. Reset: Fixed,0000\_003Fh. See Core::X86::Cpuid::CachePropEcxc0[CacheNumSets].**CPUID\_Fn8000001D\_ECX\_x02 [Cache Properties (L2)] (Core::X86::Cpuid::CachePropEcxc2)**

Read-only. Reset: Fixed,0000\_03FFh.

Core::X86::Cpuid::CachePropEcxc2 reports topology information for the L2. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_ECX\_x02

**Bits Description**31:0 **CacheNumSets: cache number of sets.** Read-only. Reset: Fixed,0000\_03FFh. See Core::X86::Cpuid::CachePropEcxc0[CacheNumSets].

**CPUID\_Fn8000001D\_ECX\_x03 [Cache Properties (L3)] (Core::X86::Cpuid::CachePropEc3)**

Read-only.

Core::X86::Cpuid::CachePropEc3 reports topology information for the L3.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_ECX\_x03

Bits	Description												
31:0	<b>CacheNumSets: cache number of sets.</b> Read-only. Reset: 0000_XXXXh. See Core::X86::Cpuid::CachePropEc0[CacheNumSets]. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0000_3 FFEh- 0000_0 000h</td><td>Reserved.</td></tr> <tr> <td>0000_3 FFFh</td><td>16384 L3 Cache Sets.</td></tr> <tr> <td>0000_7 FFEh- 0000_4 000h</td><td>Reserved.</td></tr> <tr> <td>0000_7 FFFh</td><td>32768 L3 Cache Sets.</td></tr> <tr> <td>FFFF_F FFFh- 0000_8 000h</td><td>Reserved.</td></tr> </table>	Value	Description	0000_3 FFEh- 0000_0 000h	Reserved.	0000_3 FFFh	16384 L3 Cache Sets.	0000_7 FFEh- 0000_4 000h	Reserved.	0000_7 FFFh	32768 L3 Cache Sets.	FFFF_F FFFh- 0000_8 000h	Reserved.
Value	Description												
0000_3 FFEh- 0000_0 000h	Reserved.												
0000_3 FFFh	16384 L3 Cache Sets.												
0000_7 FFEh- 0000_4 000h	Reserved.												
0000_7 FFFh	32768 L3 Cache Sets.												
FFFF_F FFFh- 0000_8 000h	Reserved.												

**CPUID\_Fn8000001D\_ECX\_x04 [Cache Properties Null] (Core::X86::Cpuid::CachePropEc4)**

Read-only. Reset: Fixed,0000\_0000h.

Core::X86::Cpuid::CachePropEc4 reports done/null. See Core::X86::Cpuid::CachePropEc0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_ECX\_x04

Bits	Description
31:0	<b>CacheNumSets.</b> Read-only. Reset: Fixed,0000_0000h. Cache number of sets.

**CPUID\_Fn8000001D\_EDX\_x00 [Cache Properties (DC)] (Core::X86::Cpuid::CachePropEdx0)**

Read-only. Reset: Fixed,0000\_0000h.

Core::X86::Cpuid::CachePropEdx0 reports topology information for the DC. See Core::X86::Cpuid::CachePropEc0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EDX\_x00

Bits	Description
31:2	Reserved.
1	<b>CacheInclusive: cache inclusive.</b> Read-only. Reset: Fixed,0. 0=Cache is not inclusive of lower cache levels. 1=Cache is inclusive of lower cache levels.
0	<b>WBINVD: Write-Back Invalidate/Invalidate.</b> Read-only. Reset: Fixed,0. 0=WBINVD/INVD invalidates all lower level caches of non-originating cores sharing this cache. 1=WBINVD/INVD not ensured to invalidate all lower level caches of non-originating cores sharing this cache.

**CPUID\_Fn8000001D\_EDX\_x01 [Cache Properties (IC)] (Core::X86::Cpuid::CachePropEdx1)**

Read-only. Reset: Fixed,0000\_0000h.

Core::X86::Cpuid::CachePropEdx1 reports topology information for the IC. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EDX\_x01

Bits	Description
31:2	Reserved.
1	<b>CacheInclusive: cache inclusive.</b> Read-only. Reset: Fixed,0. See Core::X86::Cpuid::CachePropEdx0[CacheInclusive].
0	<b>WBINVD: Write-Back Invalidate/Invalidate.</b> Read-only. Reset: Fixed,0. 0=WBINVD/INVD invalidates all lower level caches of non-originating cores sharing this cache. 1=WBINVD/INVD may not invalidate all lower level caches of non-originating cores sharing this cache. See Core::X86::Cpuid::CachePropEdx0[WBINVD].

**CPUID\_Fn8000001D\_EDX\_x02 [Cache Properties (L2)] (Core::X86::Cpuid::CachePropEdx2)**

Read-only. Reset: Fixed,0000\_0002h.

Core::X86::Cpuid::CachePropEdx2 reports topology information for the L2. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EDX\_x02

Bits	Description
31:2	Reserved.
1	<b>CacheInclusive: cache inclusive.</b> Read-only. Reset: Fixed,1. See Core::X86::Cpuid::CachePropEdx0[CacheInclusive].
0	<b>WBINVD: Write-Back Invalidate/Invalidate.</b> Read-only. Reset: Fixed,0. 0=WBINVD/INVD invalidates all lower level caches of non-originating cores sharing this cache. 1=WBINVD/INVD may not invalidate all lower level caches of non-originating cores sharing this cache.

**CPUID\_Fn8000001D\_EDX\_x03 [Cache Properties (L3)] (Core::X86::Cpuid::CachePropEdx3)**

Read-only. Reset: Fixed,0000\_0001h.

Core::X86::Cpuid::CachePropEdx3 reports reports topology information for the L3. See

Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EDX\_x03

Bits	Description
31:2	Reserved.
1	<b>CacheInclusive: cache inclusive.</b> Read-only. Reset: Fixed,0. See Core::X86::Cpuid::CachePropEdx0[CacheInclusive].
0	<b>WBINVD: Write-Back Invalidate/Invalidate.</b> Read-only. Reset: Fixed,1. 0=WBINVD/INVD invalidates all lower level caches of non-originating cores sharing this cache. 1=WBINVD/INVD may not invalidate all lower level caches of non-originating cores sharing this cache.

**CPUID\_Fn8000001D\_EDX\_x04 [Cache Properties Null] (Core::X86::Cpuid::CachePropEdx4)**

Read-only. Reset: Fixed,0000\_0000h.

Core::X86::Cpuid::CachePropEax3 reports done/null. See Core::X86::Cpuid::CachePropEax0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001D\_EDX\_x04

Bits	Description
31:0	Reserved.

**CPUID\_Fn8000001E\_EAX [Extended APIC ID] (Core::X86::Cpuid::ExtApicId)**

Read-only.

If Core::X86::Cpuid::FeatureExtIdEcX[TopologyExtensions] == 0 then CPUID Fn8000001E\_E[D,C,B,A]X are reserved.  
 If (Core::X86::Msr::APIC\_BAR[ApicEn] == 0) then Core::X86::Cpuid::ExtApicId[ExtendedApicId] is Reserved.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001E\_EAX

Bits	Description
31:0	<b>ExtendedApicId: extended APIC ID.</b> Read-only. Reserved. Reset: (Core::X86::Msr::APIC_BAR[ApicEn] && Core::X86::Msr::APIC_BAR[x2ApicEn]) ? Core::X86::Msr::APIC_ID[ApicId[31:0]] : Core::X86::Msr::APIC_BAR[ApicEn] ? {00_0000h , Core::X86::Apic::ApicId[ApicId]} : 0000_0000h.

**CPUID\_Fn8000001E\_EBX [Core Identifiers] (Core::X86::Cpuid::CoreId)**

Read-only.

See Core::X86::Cpuid::ExtApicId.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001E\_EBX

Bits	Description
31:16	Reserved.
15:8	<b>ThreadsPerCore: threads per core.</b> Read-only. Reset: XXh. The number of threads per core is ThreadsPerCore+1.
7:0	<b>CoreId: core ID.</b> Read-only. Reset: XXh. Identifies the unique per-socket logical core unit ID.

**CPUID\_Fn8000001E\_ECX [Node Identifiers] (Core::X86::Cpuid::NodeId)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001E\_ECX

Bits	Description						
31:11	Reserved.						
10:8	<b>NodesPerProcessor.</b> Read-only. Reset: XXXb. Nodes per processor. <b>ValidValues:</b>						
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>1 node per processor.</td></tr> <tr> <td>7h-1h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	1 node per processor.	7h-1h	Reserved.
Value	Description						
0h	1 node per processor.						
7h-1h	Reserved.						
7:0	<b>NodeId.</b> Read-only. Reset: Fixed,XXh. Node ID.						

**CPUID\_Fn8000001F\_EAX [AMD Secure Encryption EAX] (Core::X86::Cpuid::SecureEncryptionEax)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001F\_EAX

Bits	Description
31	Reserved.
30	<b>HvInUseWrAllowed.</b> Read-only. Reset: Fixed,0. Writes to hypervisor-owned pages are allowed when marked InUse.
29:25	Reserved.
24	<b>VmsaRegProt.</b> Read-only. Reset: Fixed,0. VMSA Register Protection is supported.
23	<b>SegmentedRmp.</b> Read-only. Reset: Fixed,0. Segmented RMP is supported.
22	<b>GuestInterceptControl.</b> Read-only. Reset: Fixed,0. Guest Intercept Control feature is supported for SEV_ES guests.
21	<b>RmpReadInstruction.</b> Read-only. Reset: Fixed,1. RMPREAD instruction is supported.
20	<b>SevFeaturesOptinPmcVirt.</b> Read-only. Reset: Fixed,0. PMC virtualization is supported for SEV-ES guests via opt-in control in SEV_FEATURES.
19	<b>IbsVirtualization.</b> Read-only. Reset: Fixed,0. IBS state virtualization is supported for SEV-ES guests via opt-in control in SEV_FEATURES.
18	<b>VirtualTomMsr.</b> Read-only. Reset: Fixed,0. Virtual TOM MSR is supported.
17	<b>VmgexitParameter.</b> Read-only. Reset: Fixed,0. VmgexitParameter is supported in SEV_FEATURES.
16	<b>VTE: Virtual Transparent Encryption for SEV.</b> Read-only. Reset: Fixed,0. The Virtual Transparent Encryption feature can be enabled to force all memory accesses within an SEV guest to be encrypted with the guest's key. When enabled the hardware pretends that the C-bits for all guest mode accesses are 1 regardless of the actual guest page tables.
15	<b>PreventHostIBS.</b> Read-only. Reset: Fixed,0. Prevent host IBS for a SEV-ES guest.
14	<b>DebugStateSwap.</b> Read-only. Reset: Fixed,0. 1=DR0-3 and DR0-3_MASK can be saved/restored on world switches.
13	<b>AlternateInjection.</b> Read-only. Reset: Fixed,0. 1=SEV-ES guests can use an encrypted vmcb field for event injection.
12	<b>RestrictInjection.</b> Read-only. Reset: Fixed,0. 1=SEV-ES guests can refuse all event-injections except #HV.
11	<b>Req64BitHypervisor.</b> Read-only. Reset: Fixed,0. Require 64-Bit Hypervisor.
10	<b>CoherencyEnforced.</b> Read-only. Reset: Fixed,0. Hardware enforces cache coherency.
9	<b>TscAuxVirtualization.</b> Read-only. Reset: Fixed,0. Hardware virtualizes TSC_AUX.
8	<b>SecureTsc.</b> Read-only. Reset: Fixed,0. Support for Secure TSC.
7	<b>VmplSSS.</b> Read-only. Reset: 0. The Supervisor Shadow Stack bit is supported in the VMPL permission set.
6	<b>RMPQUERY.</b> Read-only. Reset: 0. The RMPQUERY instruction is supported.
5	<b>VMPL.</b> Read-only. Reset: Fixed,0. VMPL VM Permission Levels supported.
4	<b>SNP.</b> Read-only. Reset: Fixed,0. RMP table can be enabled to protect memory even from hypervisor.
3	<b>SevEs.</b> Read-only. Reset: Fixed,0. Secure Encrypted ES.
2	<b>VmPgFlush: VM Page Flush MSR support.</b> Read-only. Reset: Fixed,0. VMPAGE_FLUSH is no longer supported.
1	<b>SEV.</b> Read-only. Reset: Fixed,0. Secure Encrypted Virtualization supported.
0	<b>SME.</b> Read-only. Reset: Fixed,1. Secure Memory Encryption supported.



**CPUID\_Fn8000001F\_EBX [AMD Secure Encryption EBX] (Core::X86::Cpuid::SecureEncryptionEbx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001F\_EBX

Bits	Description																		
31:16	Reserved.																		
15:12	<b>VmplSupported.</b> Read-only. Reset: Fixed, 0h. Number of VMPLs supported.																		
11:6	<b>MemEncryptPhysAddWidth.</b> Read-only. Reset: 000XXXb. Reduction of physical address space in bits when memory encryption is enabled (0 indicates no reduction). <b>Valid Values:</b>																		
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Physical Address width is not reduced.</td></tr> <tr> <td>01h</td><td>Physical Address width is reduced by one.</td></tr> <tr> <td>02h</td><td>Physical Address width is reduced by two.</td></tr> <tr> <td>03h</td><td>Physical Address width is reduced by three.</td></tr> <tr> <td>04h</td><td>Physical Address width is reduced by four.</td></tr> <tr> <td>05h</td><td>Physical Address width is reduced by five.</td></tr> <tr> <td>06h</td><td>Physical Address width is reduced by six.</td></tr> <tr> <td>3Fh-07h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Physical Address width is not reduced.	01h	Physical Address width is reduced by one.	02h	Physical Address width is reduced by two.	03h	Physical Address width is reduced by three.	04h	Physical Address width is reduced by four.	05h	Physical Address width is reduced by five.	06h	Physical Address width is reduced by six.	3Fh-07h	Reserved.
Value	Description																		
00h	Physical Address width is not reduced.																		
01h	Physical Address width is reduced by one.																		
02h	Physical Address width is reduced by two.																		
03h	Physical Address width is reduced by three.																		
04h	Physical Address width is reduced by four.																		
05h	Physical Address width is reduced by five.																		
06h	Physical Address width is reduced by six.																		
3Fh-07h	Reserved.																		
5:0	<b>CBit.</b> Read-only. Reset: 33h. Page table bit number used to enable memory encryption.																		

**CPUID\_Fn8000001F\_ECX [AMD Secure Encryption ECX] (Core::X86::Cpuid::SecureEncryptionEcx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001F\_ECX

Bits	Description
31:0	<b>NumEncryptedGuests.</b> Read-only. Reset: XXXX_XXXXh. Indicates the maximum ASID value that may be used for an SEV-enabled guest.

**CPUID\_Fn8000001F\_EDX [Minimum ASID] (Core::X86::Cpuid::SecureEncryptionEdx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn8000001F\_EDX

Bits	Description
31:0	<b>MinimumSEVASID: Minimum SEV enabled, SEV-ES disabled ASID.</b> Read-only. Reset: 0000_000Xh. Indicates the minimum ASID value that must be used for an SEV-enabled, SEV-ES-disabled guest.

**CPUID\_Fn80000020\_EAX\_x00 [Platform QoS Extended Feature Identifiers] (Core::X86::Cpuid::AmdQosExtEax0)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EAX\_x00

Bits	Description
31:0	Reserved.



**CPUID\_Fn80000020\_EBX\_x00 [Platform QoS Extended Feature Identifiers]  
(Core::X86::Cpuid::AmdQosExtEbx0)**

Read-only. Reset: 0000\_007Eh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EBX\_x00

Bits	Description
31:7	Reserved.
6	<b>CDMA_TO_MAX_CBM.</b> Read-only. Reset: 1. CDMA L3 Cache Isolation Support.
5	<b>ABMC.</b> Read-only. Reset: 1. Assignable Bandwidth Monitoring Counters..
4	<b>L3RR.</b> Read-only. Reset: 1. AMD L3 Range Reservation.
3	<b>EVT_CFG.</b> Read-only. Reset: 1. AMD Bandwidth Monitoring Event Configuration.
2	<b>SMBE.</b> Read-only. Reset: 1. Slow Memory Bandwidth Enforcement. See Core::X86::Cpuid::PqeBandwidthEax2 - Core::X86::Cpuid::PqeBandwidthEdx2.
1	<b>MBE.</b> Read-only. Reset: 1. Memory Bandwidth Enforcement.
0	Reserved.

**CPUID\_Fn80000020\_ECX\_x00 [Platform QoS Extended Feature Identifiers]  
(Core::X86::Cpuid::AmdQosExtEcx0)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_ECX\_x00

Bits	Description
31:0	Reserved.

**CPUID\_Fn80000020\_EDX\_x00 [Platform QoS Extended Feature Identifiers]  
(Core::X86::Cpuid::AmdQosExtEdx0)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EDX\_x00

Bits	Description
31:0	Reserved.

**CPUID\_Fn80000020\_EAX\_x01 [Platform QoS Enforcement for Memory Bandwidth]  
(Core::X86::Cpuid::PqeBandwidthEax1)**

Read-only. Reset: 0000\_000Ch.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EAX\_x01

Bits	Description
31:0	<b>BW_LEN: QOS Memory Bandwidth Enforcement Limit Size.</b> Read-only. Reset: 0000_000Ch. Size of the QOS Memory Bandwidth Enforcement Limit.

**CPUID\_Fn80000020\_EBX\_x01 [Platform QoS Enforcement for Memory Bandwidth]  
(Core::X86::Cpuid::PqeBandwidthEbx1)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EBX\_x01

Bits	Description
31:0	Reserved.

**CPUID\_Fn80000020\_ECX\_x01 [Platform QoS Enforcement for Memory Bandwidth]  
(Core::X86::Cpuid::PqeBandwidthEcx1)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_ECX\_x01

Bits	Description
31:0	Reserved.

**CPUID\_Fn80000020\_EDX\_x01 [Platform QoS Enforcement for Memory Bandwidth]  
(Core::X86::Cpuid::PqeBandwidthEdx1)**

Read-only. Reset: 0000\_000Fh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EDX\_x01

Bits	Description
------	-------------

31:0	<b>NumClassService.</b> Read-only. Reset: 0000_000Fh. Number of classes of service.
------	---

**CPUID\_Fn80000020\_EAX\_x02 [Enforcement for Slow Memory Bandwidth]  
(Core::X86::Cpuid::PqeBandwidthEax2)**

Read-only. Reset: 0000\_000Ch.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EAX\_x02

Bits	Description
------	-------------

31:0	<b>BW_LEN.</b> Read-only. Reset: 0000_000Ch. Slow Memory Bandwidth Enforcement Bit Range Length.
------	--

**CPUID\_Fn80000020\_EBX\_x02 [Enforcement for Slow Memory Bandwidth]  
(Core::X86::Cpuid::PqeBandwidthEbx2)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EBX\_x02

Bits	Description
------	-------------

31:0	Reserved.
------	-----------

**CPUID\_Fn80000020\_ECX\_x02 [Enforcement for Slow Memory Bandwidth]  
(Core::X86::Cpuid::PqeBandwidthEcx2)**

Read-only. Reset: 0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_ECX\_x02

Bits	Description
------	-------------

31:0	Reserved.
------	-----------

**CPUID\_Fn80000020\_EDX\_x02 [Enforcement for Slow Memory Bandwidth]  
(Core::X86::Cpuid::PqeBandwidthEdx2)**

Read-only. Reset: 0000\_000Fh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EDX\_x02

Bits	Description
------	-------------

31:0	<b>COS_MAX.</b> Read-only. Reset: 0000_000Fh. Maximum Class Of Service (COS) number for Memory Bandwidth Enforcement.
------	---

**CPUID\_Fn80000020\_EBX\_x03 [Platform QoS Monitoring Bandwidth Event Configuration]  
(Core::X86::Cpuid::PqeBandwidthEvtCfgEbx3)**

Read-only. Reset: 0000\_0002h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EBX\_x03

Bits	Description
------	-------------

31:8	Reserved.
------	-----------

7:0	<b>EVT_NUM.</b> Read-only. Reset: 02h. Reports the number of bandwidth events that can be configured.
-----	---

**CPUID\_Fn80000020\_ECX\_x03 [Platform QoS Monitoring Bandwidth Event Configuration]  
(Core::X86::Cpuid::PqeBandwidthEvtCfgEc3)**

Read-only. Reset: 0000\_007Fh.

Identifies the bandwidth sources that can be tracked.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_ECX\_x03

Bits	Description
31:7	Reserved.
6	<b>L3CacheBwVicMon.</b> Read-only. Reset: 1. Dirty victims to all types of memory.
5	<b>L3CacheRmtSlowBwFillMon.</b> Read-only. Reset: 1. Reads from remote memory the system identifies as slow memory.
4	<b>L3CacheLclSlowBwFillMon.</b> Read-only. Reset: 1. Reads from local memory the system identifies as slow memory.
3	<b>L3CacheRmtBwNtWrMon.</b> Read-only. Reset: 1. Non-temporal writes to remote memory.
2	<b>L3CacheLclBwNtWrMon.</b> Read-only. Reset: 1. Non-temporal writes to local memory.
1	<b>L3CacheRmtBwFillMon.</b> Read-only. Reset: 1. Reads from remote DRAM memory.
0	<b>L3CacheLclBwFillMon.</b> Read-only. Reset: 1. Reads from local DRAM memory.

**CPUID\_Fn80000020\_EAX\_x05 [Assignable Bandwidth Monitoring Counters]  
(Core::X86::Cpuid::PqeBandwidthEax5)**

Read-only. Reset: 0000\_0014h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EAX\_x05

Bits	Description																
31:9	Reserved.																
8	<b>OverflowBit.</b> Read-only. Reset: 0. If set, indicates whether bit 61 in QM_CTR MSR is an overflow bit.																
7:0	<b>CounterSize.</b> Read-only. Reset: 14h. Encode counter width in the QM_CTR MSR offset from 24b. Several example values are enumerated below. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Indicates that family / model stepping should be used to determine counter size.</td></tr> <tr> <td>01h</td><td>25-bit Counters.</td></tr> <tr> <td>13h-02h</td><td>Reserved.</td></tr> <tr> <td>14h</td><td>44-bit Counters.</td></tr> <tr> <td>24h-15h</td><td>Reserved.</td></tr> <tr> <td>25h</td><td>61-bit Counters.</td></tr> <tr> <td>FFh-26h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Indicates that family / model stepping should be used to determine counter size.	01h	25-bit Counters.	13h-02h	Reserved.	14h	44-bit Counters.	24h-15h	Reserved.	25h	61-bit Counters.	FFh-26h	Reserved.
Value	Description																
00h	Indicates that family / model stepping should be used to determine counter size.																
01h	25-bit Counters.																
13h-02h	Reserved.																
14h	44-bit Counters.																
24h-15h	Reserved.																
25h	61-bit Counters.																
FFh-26h	Reserved.																

**CPUID\_Fn80000020\_EBX\_x05 [Assignable Bandwidth Monitoring Counters]  
(Core::X86::Cpuid::PqeBandwidthEvtCfgEbx5)**

Read-only. Reset: 0000\_001Fh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_EBX\_x05

Bits	Description
31:16	Reserved.
15:0	<b>MAX_ABMC.</b> Read-only. Reset: 001Fh. Maximum Supported Assignable Bandwidth Monitoring Counter ID.

**CPUID\_Fn80000020\_ECX\_x05 [Assignable Bandwidth Monitoring Counters]  
(Core::X86::Cpuid::PqeBandwidthEvtCfgEc5)**

Read-only. Reset: 0000\_0001h.

Identifies the bandwidth sources that can be tracked.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000020\_ECX\_x05

Bits	Description
31:1	Reserved.
0	<b>SELECT_CLOS.</b> Read-only. Reset: 1. The user can configure individual bandwidth monitoring counters to measure Bandwidth consumed by a CLOS instead of an RMID.

**CPUID\_Fn80000021\_EAX [Extended Feature 2 EAX] (Core::X86::Cpuid::FeatureExt2Eax)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000021\_EAX

Bits	Description
31	<b>SRSO_MSR_FIX.</b> Read-only. Reset: Fixed,0. Indicates that software can set BP_CFG[4] to mitigate any other cases of Speculative Return Stack Overflow.
30	<b>SRSO_USER_KERNEL_NO.</b> Read-only. Reset: Fixed,1. Indicates that the CPU is not vulnerable to Speculative Return Stack Overflow at the user-kernel boundary.
29	<b>SRSO_NO.</b> Read-only. Reset: Fixed,0. Indicates that the CPU is not vulnerable to Speculative Return Stack Overflow.
28	<b>IBPB_BRTYPE.</b> Read-only. Reset: Fixed,1. Indicates Core::X86::Msr::PRED_CMD[IBPB] flushes all branch type predictions from the branch predictor.
27	<b>SBPB.</b> Read-only. Reset: Fixed,1. Indicates support for the Selective Branch Predictor Barrier.
26:25	Reserved.
24	<b>ERAPS.</b> Read-only. Reset: 1. Indicates support for enhanced return address predictor security.
23	Reserved.
22	<b>WL_CLASS_SUPPORT.</b> Read-only. Reset: 0. Indicates support for workload based heuristic feedback to OS for scheduling decisions.
21	<b>FP512_DOWNGRADE.</b> Read-only. Reset: 1. Indicates support for downgrading FP512 datapath to FP256.
20	<b>PREFETCHI.</b> Read-only. Reset: 1. Indicates support for IC prefetch.
19	<b>FAST_REP_SCASB.</b> Read-only. Reset: 1. Indicates support for Fast short REP SCASB.
18	<b>EPSF.</b> Read-only. Reset: 1. Enhanced Predictive Store Forwarding supported.
17	<b>GpOnUserCpuid.</b> Read-only. Reset: 1. Indicates support for #GP when executing CPUID at CPL > 0.
16	<b>OPCODE_0F017_RECLAIM.</b> Read-only. Reset: 1. Reserves opcode space 0F 01/7 for AMD use, returns illegal instruction for opcodes in this space.
15	<b>AMD_ERMSB.</b> Read-only. Reset: 1. Processor supports AMD implementation of Enhanced REP MOVSB/STOSB.
14	<b>L2TlbSizeX32.</b> Read-only. Reset: 1. Indicates that L2TLB sizes are encoded as multiples of 32.
13	<b>PrefetchCtlMsr.</b> Read-only. Reset: 1. 1=Prefetch control MSR supported. See Core::X86::Msr::PrefetchControl.
12	<b>PMC2PreciseRetire.</b> Read-only. Reset: 1. Indicates support for Core::X86::Msr::PERF_LEGACY_CTL2[PreciseRetire].
11	<b>FSRC.</b> Read-only. Reset: Fixed,1. Fast Short Repe Cmpsb supported.
10	<b>FSRS.</b> Read-only. Reset: Fixed,1. Fast Short Rep Stosb supported.
9	<b>NoSmmCtlMSR.</b> Read-only. Reset: Fixed,1. Indicates that MSRC001_0116 (SMM_CTL) is not present.
8	<b>AutomaticIBRS.</b> Read-only. Reset: Fixed,1. Indicates that Core::X86::Msr::EFER[AutomaticIBRSEn] can be set to automatically toggle IBRS protection when CPL changes.
7	<b>UpperAddressIgnore.</b> Read-only. Reset: Fixed,1. Indicates that Core::X86::Msr::EFER[UAIE] bit controls canonical check for upper address bits for certain types of accesses.
6	<b>NullSelectorClearsBase.</b> Read-only. Reset: 1. 1=Null Selector Clears Base. When this bit is set, a null segment load clears the segment base.
5:4	Reserved.
3	<b>SmmPgCfgLock.</b> Read-only. Reset: Fixed,1. 1=SMM paging configuration lock supported.
2	<b>LFenceAlwaysSerializing.</b> Read-only. Reset: Fixed,1. LFENCE is always serializing.
1	<b>FsGsKernelGsBaseNonSerializing.</b> Read-only. Reset: Fixed,1. WRMSR to Core::X86::Msr::FS_BASE, Core::X86::Msr::GS_BASE, and Core::X86::Msr::KernelGSbase are non-serializing.
0	<b>NoNestedDataBp.</b> Read-only. Reset: Fixed,1. New data-breakpoints are ignored while switching to data-breakpoint handler.

**CPUID\_Fn80000021\_EBX [Extended Feature 2 EBX] (Core::X86::Cpuid::FeatureExt2Ebx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000021\_EBX

Bits	Description
31:24	Reserved.
23:16	<b>RapSize.</b> Read-only. Reset: 08h. Return Address Predictor size. RapSize x 8 is the minimum number of CALL instructions software needs to execute to flush the RAP. Note that the CALL instructions cannot be followed by RET instructions.
15:0	<b>MicrocodePatchSize.</b> Read-only. Reset: XXXXh. Reports the size of the Microcode patch in 16-byte multiples. If 0, the size of the patch is at most 5568 (15C0h) bytes.

**CPUID\_Fn80000022\_EAX [Extended Performance Monitoring and Debug EAX] (Core::X86::Cpuid::ExtPerfMonAndDbgEax)**

Read-only. Reset: Fixed,0000\_0007h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000022\_EAX

Bits	Description
31:3	Reserved.
2	<b>LbrAndPmcFreeze.</b> Read-only. Reset: Fixed,1. 1=PMC and LBR freeze is supported in Core::X86::Msr::DBG_CTL_MSR.
1	<b>LbrExtV2.</b> Read-only. Reset: Fixed,1. 1=LBR extension Version 2 supported.
0	<b>PerfMonV2.</b> Read-only. Reset: Fixed,1. 1=Performance Monitoring Version 2 supported.

**CPUID\_Fn80000022\_EBX [Extended Performance Monitoring and Debug EBX] (Core::X86::Cpuid::ExtPerfMonAndDbgEbx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000022\_EBX

Bits	Description
31:24	Reserved.
23:16	<b>NumPerfCtrUmc.</b> Read-only. Reset: Fixed,XXXXXXXXb. Number of available UMC PMCs.
15:10	<b>NumPerfCtrNB.</b> Read-only. Reset: Fixed,08h. Number of available Data Fabric (Northbridge) Performance Monitor Counters.
9:4	<b>LbrV2StackSz.</b> Read-only. Reset: Fixed,10h. Number of available LBR stack entries.
3:0	<b>NumPerfCtrCore.</b> Read-only. Reset: Fixed,6h. Number of Core Performance Counters.

**CPUID\_Fn80000022\_ECX [Extended Performance Monitoring and Debug ECX] (Core::X86::Cpuid::ExtPerfMonAndDbgEcx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000022\_ECX

Bits	Description
31:0	<b>ActiveUmcMask.</b> Read-only. Reset: Fixed,XXXX_XXXXh. Bitmask representing active UMCs. Calculate the number of PMCs per UMC as Core::X86::Cpuid::ExtPerfMonAndDbgEbx[NumPerfCtrUmc] / POPCNT(CPUID_8000_0022_ECX[31:0]).

**CPUID\_Fn80000023\_EAX [AMD Secure Multi-Key Encryption EAX] (Core::X86::Cpuid::SecureMultiKeyEncryptionEax)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000023\_EAX

Bits	Description
31:1	Reserved.
0	<b>MemHmkSupport.</b> Read-only. Reset: Fixed,0. 1=MEM-HMK mode is supported.

**CPUID\_Fn80000023\_EBX [AMD Secure Multi-Key Encryption EBX]  
(Core::X86::Cpuid::SecureMultiKeyEncryptionEbx)**

Read-only. Reset: Fixed,0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000023\_EBX

Bits	Description
31:16	Reserved.
15:0	<b>MaxMemHmkEncrKeyID.</b> Read-only. Reset: Fixed,0000h. Total number of available encryption keys in MEM-HMK mode.

**CPUID\_Fn80000023\_ECX [AMD Secure Multi-Key Encryption ECX]  
(Core::X86::Cpuid::SecureMultiKeyEncryptionEcx)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000023\_ECX

Bits	Description
31:0	Reserved.

**CPUID\_Fn80000023\_EDX [AMD Secure Multi-Key Encryption EDX]  
(Core::X86::Cpuid::SecureMultiKeyEncryptionEdx)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000023\_EDX

Bits	Description
31:0	Reserved.

**CPUID\_Fn80000026\_EAX\_x0[0...3] [Extended CPU Topology] (Core::X86::Cpuid::ExCpuTopologyEax)**

Read-only.

CPUID Fn8000\_0026\_E[D,C,B,A]X\_x[3:0] specifies the hierarchy of logical cores from the SMT level through the processor socket level. Software reads CPUID Fn8000\_0026\_E[C,B,A]X for ascending values of ECX until (CPUID Fn8000\_0026\_EBX[LogProcAtThisLevel] == 0).

Note: While CPUID Fn8000\_0026 is a preferred superset to CPUID\_Fn0000000B, CPUID\_Fn0000000B information is valid for software for the supported levels on AMD.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; CPUID\_Fn80000026\_EAX\_x00

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; CPUID\_Fn80000026\_EAX\_x01

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; CPUID\_Fn80000026\_EAX\_x02

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; CPUID\_Fn80000026\_EAX\_x03

Bits	Description
31	<b>AsymmetricCores.</b> Read-only. Reset: Fixed,X. When set, not all cores in the system report the same value in Core::X86::Cpuid::ExtCpuTopologyEbx[LogProcThisLevel] for Core::X86::Cpuid::ExtCpuTopologyEcx[LevelType]=0x1. When cleared, all cores in the system report the same value in Core::X86::Cpuid::ExtCpuTopologyEbx[LogProcThisLevel] for Core::X86::Cpuid::ExtCpuTopologyEcx[LevelType]=0x1.
30	<b>HeterogeneousCoreTopology.</b> Read-only. Reset: Fixed,0. When set, not all instances at the current hierarchy level have the same Core Type topology. The core type is reported in Core::X86::Cpuid::ExtCpuTopologyEbx[CoreType].
29	<b>EfficiencyRankingAvailable.</b> Read-only. Reset: Fixed,0. When set, Core::X86::Cpuid::ExtCpuTopologyEbx[ProcessorPowerEfficiencyRanking] is valid and varies between individual cores.
28:5	Reserved.
4:0	<b>CoreMaskWidth.</b> Read-only. Reset: Fixed,XXh. Number of bits to shift Core::X86::Cpuid::ExtCpuTopologyEdx[ExtendedLocalApicId] right to get unique topology ID of the next instance of the current level type. All logical processors with the same next level ID share current level.



**CPUID\_Fn80000026\_EBX\_x0[0...3] [Extended CPU Topology] (Core::X86::Cpuid::ExtCpuTopologyEbx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; CPUID\_Fn80000026\_EBX\_x00

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; CPUID\_Fn80000026\_EBX\_x01

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; CPUID\_Fn80000026\_EBX\_x02

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; CPUID\_Fn80000026\_EBX\_x03

Bits	Description								
31:28	<b>CoreType.</b> Read-only. Reset: Fixed,Xh. Defines per-core architectural feature differentiation (microarchitectural resources, etc.) that may lead to a different performance, core clock boost, and power characteristic. Only valid while LevelType=Core. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Performance Core.</td></tr> <tr> <td>1h</td><td>Efficiency Core.</td></tr> <tr> <td>Fh-2h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	Performance Core.	1h	Efficiency Core.	Fh-2h	Reserved.
Value	Description								
0h	Performance Core.								
1h	Efficiency Core.								
Fh-2h	Reserved.								
27:24	<b>NativeModelId.</b> Read-only. Reset: Fixed,0h. Context sensitive to CPUID_Fn00000001_EAX and CPUID_Fn80000001_EAX [Family, Model, Stepping Identifiers] (Core::X86::Cpuid::FamModStep). Only valid while Level Type=Core. This value is used by software to further differentiate implementation specific software visible properties between the cores. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Zen5 Core.</td></tr> <tr> <td>Fh-1h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	Zen5 Core.	Fh-1h	Reserved.		
Value	Description								
0h	Zen5 Core.								
Fh-1h	Reserved.								
23:16	<b>ProcessorPowerEfficiencyRanking.</b> Read-only. Reset: Fixed,XXh. Identifies a static efficiency ranking between each of the cores of a specific CoreType, where a core with a lower value has intrinsically better power, but potentially lower performance potential vs cores with a higher value.								
15:0	<b>LogProcThisLevel.</b> Read-only. Reset: Fixed,XXXh. Number of logical processors at this level type.								

**CPUID\_Fn80000026\_ECX\_x0[0...3] [Extended CPU Topology] (Core::X86::Cpuid::ExtCpuTopologyEcX)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; CPUID\_Fn80000026\_ECX\_x00

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; CPUID\_Fn80000026\_ECX\_x01

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; CPUID\_Fn80000026\_ECX\_x02

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; CPUID\_Fn80000026\_ECX\_x03

Bits	Description														
31:16	Reserved.														
15:8	<b>LevelType.</b> Read-only. Reset: Fixed,XXh. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>01h</td><td>Core</td></tr> <tr> <td>02h</td><td>Complex</td></tr> <tr> <td>03h</td><td>Reserved.</td></tr> <tr> <td>04h</td><td>Socket</td></tr> <tr> <td>FFh-05h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Reserved.	01h	Core	02h	Complex	03h	Reserved.	04h	Socket	FFh-05h	Reserved.
Value	Description														
00h	Reserved.														
01h	Core														
02h	Complex														
03h	Reserved.														
04h	Socket														
FFh-05h	Reserved.														
7:0	<b>EcXVal.</b> Read-only. Reset: Fixed,XXh. ECX input value.														



**CPUID\_Fn80000026\_EDX [Extended CPU Topology] (Core::X86::Cpuid::ExtCpuTopologyEdx)**

Read-only.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; CPUID\_Fn80000026\_EDX

**Bits Description**31:0 **ExtendedLocalApicId.** Read-only. Reset: Fixed,XXXX\_XXXXh. Extended APIC ID.**2.1.13 MSR Registers****2.1.13.1 MSRs - MSR0000\_xxxx****MSR0000\_0010 [Time Stamp Counter] (Core::X86::Msr::TSC)**

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

The TSC uses a common reference for all sockets, cores and threads.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0010

**Bits Description**63:0 **TSC: time stamp counter.** Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A Read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the TSC P0 frequency based value (as if TSCRatio == 1.0) when (TSCRatio != 1.0).**MSR0000\_001B [APIC Base Address] (Core::X86::Msr::APIC\_BAR)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_001B

**Bits Description**

63:48 Reserved.

47:12 **ApicBar[47:12]: APIC base address register.** Read-write. Reset: 0\_000F\_EE00h. Specifies the base address, physical address [47:12], for the APICXX register set in xAPIC mode. See 2.1.11.2.1.2 [APIC Register Space].11 **ApicEn: APIC enable.** Read-write. Reset: 0. 0=Disable Local APIC. 1=Local APIC is enabled in xAPIC mode. See 2.1.11.2.1.2 [APIC Register Space].10 **x2ApicEn: Extended APIC enable.** Read-write. Reset: 0. 0=Disable Extended Local APIC. 1=Extended Local APIC is enabled in x2APIC mode. Clearing this bit after it has been set requires ApicEn to be cleared as well.

9 Reserved.

8 **BSC: boot strap core.** Read-write, Volatile. Reset: X. 0=The core is not the boot core of the BSP. 1=The core is the boot core of the BSP.

7:0 Reserved.

**MSR0000\_002A [Cluster ID] (Core::X86::Msr::EBL\_CR\_POWERON)**

Writes to this register result in a GP fault with error code 0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_002A

**Bits Description**

63:18 Reserved.

17:16 **ClusterID.** Read, Error-on-write. Reset: 0h. The field does not affect hardware.

15:0 Reserved.

**MSR0000\_003B [Time Stamp Counter Adjustment] (Core::X86::Msr::TSC\_ADJUST)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_003B

**Bits Description**63:0 **TimeStampCounterAdjustment.** Read-write. Reset: 0000\_0000\_0000\_0000h. Provides an offset to the TSC when the TSC is read. This value changes with a write, and does not change as time elapses.

**MSR0000\_0048 [Speculative Control] (Core::X86::Msr::SPEC\_CTRL)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0048

Bits	Description
63:8	Reserved.
7	<b>PSFD: Predictive Store Forwarding Disable.</b> Read-write. Reset: 0. 1=Disable predictive store forwarding.
6:3	Reserved.
2	<b>SSBD.</b> Read-write. Reset: 0. Speculative Store Bypass Disable.
1	<b>STIBP.</b> Read-write. Reset: 0. Single thread indirect branch predictor.
0	<b>IBRS.</b> Read-write. Reset: 0. Indirect branch restriction speculation.

**MSR0000\_0049 [Prediction Command] (Core::X86::Msr::PRED\_CMD)**

Write-only, Error-on-read. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]; MSR0000\_0049

Bits	Description
63:8	Reserved.
7	<b>SBPB: selective branch predictor barrier.</b> Write-only, Error-on-read. Reset: 0. When SBPB is supported (Core::X86::Cpuid::FeatureExt2Eax[SBPB]==1), setting this bit initiates a selective branch predictor barrier
6:1	Reserved.
0	<b>IBPB: indirect branch prediction barrier.</b> Write-only, Error-on-read. Reset: 0. Supported if Core::X86::Cpuid::FeatureExtIdEbx[IBPB] == 1.

**MSR0000\_008B [Patch Level] (Core::X86::Msr::PATCH\_LEVEL)**

Read, Error-on-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]; MSR0000\_008B

Bits	Description
63:32	Reserved.
31:0	<b>PatchLevel.</b> Read, Error-on-write, Volatile. Reset: 0000_0000h. This returns an identification number for the microcode patch that has been loaded. If no patch has been loaded, this returns 0.

**MSR0000\_00E7 [Max Performance Frequency Clock Count] (Core::X86::Msr::MPERF)**

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_00E7

Bits	Description
63:0	<b>MPERF: maximum core clocks counter.</b> Read-write, Volatile. Reset: 0000_0000_0000_0000h. Incremented by hardware at the P0 frequency while the core is in C0. This register does not increment when the core is in the stop-grant state. In combination with Core::X86::Msr::APERF, this is used to determine the effective frequency of the core. A Read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. This field uses software P-state numbering. See Core::X86::Msr::HWCR[EffFreqCntMwait], 2.1.4 [Effective Frequency]

**MSR0000\_00E8 [Actual Performance Frequency Clock Count] (Core::X86::Msr::APERF)**

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_00E8

Bits	Description
63:0	<b>APERF: actual core clocks counter.</b> Read-write, Volatile. Reset: 0000_0000_0000_0000h. This register increments in proportion to the actual number of core clocks cycles while the core is in C0. The register does not increment when the core is in the stop-grant state. See Core::X86::Msr::MPERF.

**MSR0000\_00FE [MTRR Capabilities] (Core::X86::Msr::MTRRcap)**

Read,Error-on-write. Reset: 0000\_0000\_0000\_0508h.

\_ccd[1:0]\_lthree0\_core[7:0]; MSR0000\_00FE

Bits	Description
63:11	Reserved.
10	<b>MtrrCapWc: write-combining memory type.</b> Read,Error-on-write. Reset: 1. 1=The write combining memory type is supported.
9	Reserved.
8	<b>MtrrCapFix: fixed range register.</b> Read,Error-on-write. Reset: 1. 1=Fixed MTRRs are supported.
7:0	<b>MtrrCapVCnt: variable range registers count.</b> Read,Error-on-write. Reset: 08h. Specifies the number of variable MTRRs supported.

**MSR0000\_010B [Flush Command] (Core::X86::Msr::FLUSH\_CMD)**

Writes to this register do not execute until all prior instructions finished execution and have completed locally. Later instructions do not execute until the Write completes.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_010B

Bits	Description
63:1	Reserved.
0	<b>L1D_FLUSH.</b> Write-only, Volatile. Reset: 0. When written to 1, performs a write-back and invalidate of the L1 data cache.

**MSR0000\_0174 [SYSENTER CS] (Core::X86::Msr::SYSENTER\_CS)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0174

Bits	Description
63:16	Reserved.
15:0	<b>SysEnterCS: SYSENTER target CS.</b> Read-write. Reset: 0000h. Holds the called procedure code segment.

**MSR0000\_0175 [SYSENTER ESP] (Core::X86::Msr::SYSENTER\_ESP)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0175

Bits	Description
63:32	Reserved.
31:0	<b>SysEnterESP: SYSENTER target SP.</b> Read-write. Reset: 0000_0000h. Holds the called procedure stack pointer.

**MSR0000\_0176 [SYSENTER EIP] (Core::X86::Msr::SYSENTER\_EIP)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0176

Bits	Description
63:32	Reserved.
31:0	<b>SysEnterEIP: SYSENTER target IP.</b> Read-write. Reset: 0000_0000h. Holds the called procedure instruction pointer.

**MSR0000\_0179 [Global Machine Check Capabilities] (Core::X86::Msr::MCG\_CAP)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0179

Bits	Description
63:9	Reserved.
8	<b>McgCtlP: MCG_CTL register present.</b> Read-only, Error-on-write. Reset: Fixed, 1. 1=The machine check control registers (MCI_CTL) are present. See 3.1 [Machine Check Architecture].
7:0	<b>Count.</b> Read-only, Error-on-write, Volatile. Reset: XXh. Indicates the number of error reporting banks visible to the core. This value may differ from core to core.

**MSR0000\_017A [Global Machine Check Status] (Core::X86::Msr::MCG\_STAT)**

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

See 3.1 [Machine Check Architecture].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_017A

Bits	Description
63:3	Reserved.
2	<b>MCIP</b> . Read-write, Volatile. Reset: 0. 1=A machine check is in progress. Machine check in progress.
1	<b>EIPV: error instruction pointer valid</b> . Read-write, Volatile. Reset: 0. 1=The instruction pointer that was pushed onto the stack by the machine check mechanism references the instruction that caused the machine check error.
0	<b>RIPV: restart instruction pointer valid</b> . Read-write, Volatile. Reset: 0. 0=The interrupt was not precise and/or the process (task) context may be corrupt; continued operation of this process may not be possible without intervention, however system processing or other processes may be able to continue with appropriate software clean up. 1=Program execution can be reliably restarted at the EIP address on the stack.

**MSR0000\_017B [Global Machine Check Exception Reporting Control] (Core::X86::Msr::MCG\_CTL)**

Reset: 0000\_0000\_0000\_0000h.

This register controls enablement of the individual error reporting banks; see 3.1 [Machine Check Architecture] and 3.1.2.1 [Global Registers]. When a machine check register bank is not enabled in MCG\_CTL, errors for that bank are not logged or reported, and actions enabled through the MCA are not taken; each MCi\_CTL register identifies which errors are still corrected when MCG\_CTL[i] is disabled.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_017B

Bits	Description																
63:7	<b>MCnEn</b> . Configurable. Reset: 000_0000_0000_0000h. <b>Description:</b> 1=The MC machine check register bank is enabled. Width of this field is SOC implementation and configuration specific. See 3.1.2.1 [Global Registers].																
6:0	<b>MCnEnCore</b> . Read-write. Reset: 00h. 1=The MC machine check register bank is enabled. <b>ValidValues:</b> <table> <tr> <th>Bit</th><th>Description</th></tr> <tr> <td>[0]</td><td>Enable MCA for LSDC</td></tr> <tr> <td>[1]</td><td>Enable MCA for ICBP</td></tr> <tr> <td>[2]</td><td>Enable MCA for L2</td></tr> <tr> <td>[3]</td><td>Enable MCA for DE</td></tr> <tr> <td>[4]</td><td>Reserved.</td></tr> <tr> <td>[5]</td><td>Enable MCA for SCEX</td></tr> <tr> <td>[6]</td><td>Enable MCA for FP</td></tr> </table>	Bit	Description	[0]	Enable MCA for LSDC	[1]	Enable MCA for ICBP	[2]	Enable MCA for L2	[3]	Enable MCA for DE	[4]	Reserved.	[5]	Enable MCA for SCEX	[6]	Enable MCA for FP
Bit	Description																
[0]	Enable MCA for LSDC																
[1]	Enable MCA for ICBP																
[2]	Enable MCA for L2																
[3]	Enable MCA for DE																
[4]	Reserved.																
[5]	Enable MCA for SCEX																
[6]	Enable MCA for FP																

**MSR0000\_01D9 [Debug Control] (Core::X86::Msr::DBG\_CTL\_MSR)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_01D9

Bits	Description
63:13	Reserved.
12	<b>FPMCI</b> . Read-write. Reset: 0. 1=Freeze PMCs on PMC interrupt.
11	<b>FLBRI</b> . Read-write. Reset: 0. 1=Freeze LBR Stack on PMC interrupt.
10:6	Reserved.
5:3	<b>PB: performance monitor pin control</b> . Read-write. Reset: 0h. Performance Monitor Pins are not supported on this processor.
2	<b>BusLockTrapEn</b> . Read-write. Reset: 0.
1	<b>BTF</b> . Read-write. Reset: 0. 1=Enable branch single step.
0	<b>LBR</b> . Read-write. Reset: 0. 1=Enable last branch record.

**MSR0000\_01DB [Last Branch From IP] (Core::X86::Msr::BR\_FROM)**

Read,Error-on-write,Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_01DB

Bits	Description
63:58	Reserved.
57:0	<b>LastBranchFromIP</b> . Read,Error-on-write,Volatile. Reset: 000_0000_0000_0000h. Loaded with the segment offset of the branch instruction.

**MSR0000\_01DC [Last Branch To IP] (Core::X86::Msr::BR\_TO)**

Read,Error-on-write,Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_01DC

Bits	Description
63:58	Reserved.
57:0	<b>LastBranchToIP</b> . Read,Error-on-write,Volatile. Reset: 000_0000_0000_0000h. Holds the target RIP of the last branch that occurred before an exception or interrupt.

**MSR0000\_01DD [Last Exception From IP] (Core::X86::Msr::LastExcpFromIp)**

Read,Error-on-write,Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_01DD

Bits	Description
63:58	Reserved.
57:0	<b>LastIntFromIP</b> . Read,Error-on-write,Volatile. Reset: 000_0000_0000_0000h. Holds the source RIP of the last branch that occurred before the exception or interrupt.

**MSR0000\_01DE [Last Exception To IP] (Core::X86::Msr::LastExcpToIp)**

Read,Error-on-write,Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_01DE

Bits	Description
63:58	Reserved.
57:0	<b>LastIntToIP</b> . Read,Error-on-write,Volatile. Reset: 000_0000_0000_0000h. Holds the target RIP of the last branch that occurred before the exception or interrupt.

**MSR0000\_020[0...E] [Variable-Size MTRRs Base] (Core::X86::Msrr::MtrrVarBase)**

Each MTRR (Core::X86::Msrr::MtrrVarBase, Core::X86::Msrr::MtrrFix\_64K through Core::X86::Msrr::MtrrFix\_4K\_7, or Core::X86::Msrr::MTRRdefType) specifies a physical address range and a corresponding memory type (MemType) associated with that range. Setting the memory type to an unsupported value will result in a #GP.

The variable-size MTRRs come in pairs of base and mask registers (MSR0000\_0200 and MSR0000\_0201 are the first pair, etc.). Variables MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeEn]. A core access--with address CPUAddr--is determined to be within the address range of a variable-size MTRR if the following equation is true:

$\text{CPUAddr}[47:12] \& \text{PhyMask}[47:12] == \text{PhyBase}[47:12] \& \text{PhyMask}[47:12]$ .

For example, if the variable MTRR spans 256 KB and starts at the 1 MB address the PhyBase would be set to 0\_0010\_0000h and the PhyMask to F\_FFFC\_0000h (with zeros filling in for bits[11:0]). This results in a range from 0\_0010\_0000h to 0\_0013\_FFFFh.

\_ccd[1:0]\_lthree0\_core[7:0]\_n0; MSR0000\_0200

\_ccd[1:0]\_lthree0\_core[7:0]\_n1; MSR0000\_0202

\_ccd[1:0]\_lthree0\_core[7:0]\_n2; MSR0000\_0204

\_ccd[1:0]\_lthree0\_core[7:0]\_n3; MSR0000\_0206

\_ccd[1:0]\_lthree0\_core[7:0]\_n4; MSR0000\_0208

\_ccd[1:0]\_lthree0\_core[7:0]\_n5; MSR0000\_020A

\_ccd[1:0]\_lthree0\_core[7:0]\_n6; MSR0000\_020C

\_ccd[1:0]\_lthree0\_core[7:0]\_n7; MSR0000\_020E

Bits	Description
63:48	Reserved.
47:12	<b>PhyBase: base address.</b> Read-write. Reset: X_XXXX_XXXXh. Physical base address.
11:3	Reserved.
2:0	<b>MemType: memory type.</b> Read-write. Reset: XXXb. Address range from 00000h to 0FFFFh.
<b>ValidValues:</b>	
Value	Description
0h	UC or uncacheable.
1h	WC or write combining.
3h-2h	Reserved.
4h	WT or write through.
5h	WP or write protect.
6h	WB or write back.
7h	Reserved.

**MSR0000\_020[1...F] [Variable-Size MTRRs Mask] (Core::X86::Msrr::MtrrVarMask)**

\_ccd[1:0]\_lthree0\_core[7:0]\_n0; MSR0000\_0201

\_ccd[1:0]\_lthree0\_core[7:0]\_n1; MSR0000\_0203

\_ccd[1:0]\_lthree0\_core[7:0]\_n2; MSR0000\_0205

\_ccd[1:0]\_lthree0\_core[7:0]\_n3; MSR0000\_0207

\_ccd[1:0]\_lthree0\_core[7:0]\_n4; MSR0000\_0209

\_ccd[1:0]\_lthree0\_core[7:0]\_n5; MSR0000\_020B

\_ccd[1:0]\_lthree0\_core[7:0]\_n6; MSR0000\_020D

\_ccd[1:0]\_lthree0\_core[7:0]\_n7; MSR0000\_020F

Bits	Description
63:48	Reserved.
47:12	<b>PhyMask: address mask.</b> Read-write. Reset: X_XXXX_XXXXh. Physical address mask.
11	<b>Valid: valid.</b> Read-write. Reset: X. 1=The variable-size MTRR pair is enabled.
10:0	Reserved.

**MSR0000\_0250 [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_64K)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE64K; MSR0000\_0250

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_64K_70000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_64K_70000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_64K_70000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_64K_60000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_64K_60000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_64K_60000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_64K_50000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																



	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_64K_50000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_64K_50000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_64K_40000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_64K_40000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_64K_40000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_64K_30000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_64K_30000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_64K_30000: memory type.</b> Read-write. Reset: XXXb.																



	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_64K_20000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_64K_20000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_64K_20000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_64K_10000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_64K_10000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_64K_10000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_64K_00000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from 00000h to 0FFFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_64K_00000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from 00000h to 0FFFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_64K_00000: memory type.</b> Read-write. Reset: XXXb. Address range from 00000h to 0FFFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_0258 [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_16K\_0)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE16K0; MSR0000\_0258

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_16K_9C000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_16K_9C000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_16K_9C000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_16K_98000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_16K_98000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_16K_98000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_16K_94000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_16K_94000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_16K_94000: memory type.</b> Read-write. Reset: XXXb.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_16K_90000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_16K_90000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_16K_90000: memory type.</b> Read-write. Reset: XXXb.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_16K_8C000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_16K_8C000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_16K_8C000: memory type.</b> Read-write. Reset: XXXb.																

	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_16K_88000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_16K_88000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_16K_88000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_16K_84000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_16K_84000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_16K_84000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_16K_80000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from 80000h to 83FFFh. Core::X86::Msrr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks reads of the stored value.																
	AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_16K_80000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from 80000h to 83FFFh. Core::X86::Msrr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks reads of the stored value.																
	AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_16K_80000: memory type.</b> Read-write. Reset: XXXb. Address range from 80000h to 83FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_0259 [Fixed-Size MTRRs] (Core::X86::Msr::MtrrFix\_16K\_1)**

See Core::X86::Msr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing reserved MemType values causes an error-on-write. If Core::X86::Msr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE16K1; MSR0000\_0259

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_16K_BC000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_16K_BC000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_16K_BC000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_16K_B8000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_16K_B8000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_16K_B8000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_16K_B4000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																



	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_16K_B4000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_16K_B4000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_16K_B0000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_16K_B0000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_16K_B0000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_16K_AC000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_16K_AC000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_16K_AC000: memory type.</b> Read-write. Reset: XXXb.																



	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_16K_A8000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_16K_A8000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_16K_A8000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_16K_A4000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_16K_A4000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_16K_A4000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_16K_A0000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from A0000h to A3FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_16K_A0000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from A0000h to A3FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_16K_A0000: memory type.</b> Read-write. Reset: XXXb. Address range from A0000h to A3FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_0268 [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_4K\_0)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing Reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE4K0; MSR0000\_0268

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_4K_C7000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_4K_C7000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_4K_C7000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_4K_C6000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_4K_C6000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_4K_C6000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_4K_C5000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_4K_C5000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_4K_C5000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_4K_C4000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_4K_C4000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_4K_C4000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_4K_C3000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_4K_C3000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_4K_C3000: memory type.</b> Read-write. Reset: XXXb.																

	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_4K_C2000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_4K_C2000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_4K_C2000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_4K_C1000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_4K_C1000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_4K_C1000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_4K_C0000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from C0000h to C0FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_4K_C0000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from C0000h to C0FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_4K_C0000: memory type.</b> Read-write. Reset: XXXb. Address range from C0000h to C0FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_0269 [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_4K\_1)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing Reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE4K1; MSR0000\_0269

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_4K_CF000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_4K_CF000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_4K_CF000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_4K_CE000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_4K_CE000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_4K_CE000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_4K_CD000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_4K_CD000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_4K_CD000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_4K_CC000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_4K_CC000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_4K_CC000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_4K_CB000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_4K_CB000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_4K_CB000: memory type.</b> Read-write. Reset: XXXb.																



	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_4K_CA000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_4K_CA000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_4K_CA000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_4K_C9000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_4K_C9000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_4K_C9000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_4K_C8000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from C8000 to C8FFF. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_4K_C8000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from C8000 to C8FFF. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_4K_C8000: memory type.</b> Read-write. Reset: XXXb. Address range from C8000 to C8FFF.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_026A [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_4K\_2)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing Reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE4K2; MSR0000\_026A

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_4K_D7000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_4K_D7000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_4K_D7000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_4K_D6000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_4K_D6000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_4K_D6000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_4K_D5000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_4K_D5000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_4K_D5000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_4K_D4000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_4K_D4000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_4K_D4000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_4K_D3000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_4K_D3000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_4K_D3000: memory type.</b> Read-write. Reset: XXXb.																

	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_4K_D2000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_4K_D2000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_4K_D2000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_4K_D1000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_4K_D1000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_4K_D1000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_4K_D0000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from D0000h to D0FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_4K_D0000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from D0000h to D0FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_4K_D0000: memory type.</b> Read-write. Reset: XXXb. Address range from D0000h to D0FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_026B [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_4K\_3)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing Reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE4K3; MSR0000\_026B

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_4K_DF000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_4K_DF000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_4K_DF000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_4K_DE000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_4K_DE000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_4K_DE000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_4K_DD000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_4K_DD000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_4K_DD000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_4K_DC000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_4K_DC000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_4K_DC000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_4K_DB000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_4K_DB000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_4K_DB000: memory type.</b> Read-write. Reset: XXXb.																



	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_4K_DA000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_4K_DA000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_4K_DA000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_4K_D9000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_4K_D9000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_4K_D9000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_4K_D8000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from D8000h to D8FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_4K_D8000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from D8000h to D8FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_4K_D8000: memory type.</b> Read-write. Reset: XXXb. Address range from D8000h to D8FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
	5h	WP or write protect.															
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_026C [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_4K\_4)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE4K4; MSR0000\_026C

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_4K_E7000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_4K_E7000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_4K_E7000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_4K_E6000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_4K_E6000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_4K_E6000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_4K_E5000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_4K_E5000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_4K_E5000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_4K_E4000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_4K_E4000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_4K_E4000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_4K_E3000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_4K_E3000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_4K_E3000: memory type.</b> Read-write. Reset: XXXb.																

	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_4K_E2000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_4K_E2000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_4K_E2000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_4K_E1000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_4K_E1000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_4K_E1000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_4K_E0000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from E0000h to E0FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_4K_E0000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from E0000h to E0FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_4K_E0000: memory type.</b> Read-write. Reset: XXXb. Address range from E0000h to E0FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
	5h	WP or write protect.															
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_026D [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_4K\_5)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing Reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE4K5; MSR0000\_026D

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_4K_EF000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_4K_EF000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_4K_EF000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_4K_EE000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_4K_EE000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_4K_EE000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_4K_ED000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_4K_ED000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_4K_ED000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_4K_EC000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_4K_EC000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_4K_EC000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_4K_EB000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_4K_EB000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_4K_EB000: memory type.</b> Read-write. Reset: XXXb.																



	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_4K_EA000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_4K_EA000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_4K_EA000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_4K_E9000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_4K_E9000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_4K_E9000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_4K_E8000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from E8000h to E8FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_4K_E8000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from E8000h to E8FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_4K_E8000: memory type.</b> Read-write. Reset: XXXb. Address range from E8000h to E8FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
	5h	WP or write protect.															
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_026E [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_4K\_6)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing Reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE4K6; MSR0000\_026E

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_4K_F7000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_4K_F7000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_4K_F7000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_4K_F6000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_4K_F6000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_4K_F6000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_4K_F5000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_4K_F5000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_4K_F5000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_4K_F4000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_4K_F4000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_4K_F4000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_4K_F3000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_4K_F3000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_4K_F3000: memory type.</b> Read-write. Reset: XXXb.																

	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_4K_F2000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_4K_F2000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_4K_F2000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_4K_F1000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_4K_F1000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_4K_F1000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_4K_F0000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from F0000h to F0FFF. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_4K_F0000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from F0000h to F0FFF. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_4K_F0000: memory type.</b> Read-write. Reset: XXXb. Address range from F0000h to F0FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_026F [Fixed-Size MTRRs] (Core::X86::Msrr::MtrrFix\_4K\_7)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information. Fixed MTRRs are enabled through Core::X86::Msrr::MTRRdefType[MtrrDefTypeFixEn,MtrrDefTypeEn]. For addresses below 1MB, the appropriate Fixed MTRRs override the default access destination. Each fixed MTRR includes two bits, RdDram and WrDram, that determine the destination based on the access type. Writing Reserved MemType values causes an error-on-write. If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_nSIZE4K7; MSR0000\_026F

Bits	Description																
63:61	Reserved.																
60	<b>RdDram_4K_FF000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
59	<b>WrDram_4K_FF000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
58:56	<b>MemType_4K_FF000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:53	Reserved.																
52	<b>RdDram_4K_FE000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
51	<b>WrDram_4K_FE000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msrr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
50:48	<b>MemType_4K_FE000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
47:45	Reserved.																
44	<b>RdDram_4K_FD000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.																

	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
43	<b>WrDram_4K_FD000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
42:40	<b>MemType_4K_FD000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:37	Reserved.																
36	<b>RdDram_4K_FC000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
35	<b>WrDram_4K_FC000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
34:32	<b>MemType_4K_FC000: memory type.</b> Read-write. Reset: XXXb. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
31:29	Reserved.																
28	<b>RdDram_4K_FB000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
27	<b>WrDram_4K_FB000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
26:24	<b>MemType_4K_FB000: memory type.</b> Read-write. Reset: XXXb.																



	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
23:21	Reserved.	
20	<b>RdDram_4K_FA000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
19	<b>WrDram_4K_FA000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
18:16	<b>MemType_4K_FA000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
15:13	Reserved.	
12	<b>RdDram_4K_F9000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
11	<b>WrDram_4K_F9000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM.	
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.	
10:8	<b>MemType_4K_F9000: memory type.</b> Read-write. Reset: XXXb.	
	<b>ValidValues:</b>	
	<b>Value</b>	<b>Description</b>
	0h	UC or uncacheable.
	1h	WC or write combining.
	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
7:5	Reserved.	

4	<b>RdDram_4K_F8000: read DRAM.</b> 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the range are marked as destined for DRAM. Address range from F8000h to F8FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
3	<b>WrDram_4K_F8000: write DRAM.</b> 0=Write accesses to the range are marked as MMIO. 1=Write accesses to the range are marked as destined for DRAM. Address range from F8000h to F8FFFh. Core::X86::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks Reads of the stored value.																
	AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.																
2:0	<b>MemType_4K_F8000: memory type.</b> Read-write. Reset: XXXb. Address range from F8000h to F8FFFh.																
	<b>ValidValues:</b>																
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0h</td><td>UC or uncacheable.</td></tr><tr><td>1h</td><td>WC or write combining.</td></tr><tr><td>3h-2h</td><td>Reserved.</td></tr><tr><td>4h</td><td>WT or write through.</td></tr><tr><td>5h</td><td>WP or write protect.</td></tr><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
	Value	Description															
	0h	UC or uncacheable.															
	1h	WC or write combining.															
	3h-2h	Reserved.															
	4h	WT or write through.															
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_0277 [Page Attribute Table] (Core::X86::Msr::PAT)**

This register specifies the memory type based on the PAT, PCD, and PWT bits in the virtual address page tables.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0277

Bits	Description																
63:59	Reserved.																
58:56	<b>PA7MemType.</b> Read-write. Reset: 0h. Default UC. MemType for {PAT, PCD, PWT} = 7h.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
55:51	Reserved.																
50:48	<b>PA6MemType.</b> Read-write. Reset: 7h. Default UC. MemType for {PAT, PCD, PWT} = 6h.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>UC minus.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	UC minus.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	UC minus.																
47:43	Reserved.																
42:40	<b>PA5MemType.</b> Read-write. Reset: 4h. Default WT. MemType for {PAT, PCD, PWT} = 5h.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
39:35	Reserved.																
34:32	<b>PA4MemType.</b> Read-write. Reset: 6h. Default WB. MemType for {PAT, PCD, PWT} = 4h.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

31:27	Reserved.																
26:24	<b>PA3MemType.</b> Read-write. Reset: 0h. Default UC. MemType for {PAT, PCD, PWT} = 3h.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
23:19	Reserved.																
18:16	<b>PA2MemType.</b> Read-write. Reset: 7h. Default UC. MemType for {PAT, PCD, PWT} = 2h.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>UC minus.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	UC minus.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	UC minus.																
15:11	Reserved.																
10:8	<b>PA1MemType.</b> Read-write. Reset: 4h. Default WT. MemType for {PAT, PCD, PWT} = 1h.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
7:3	Reserved.																
2:0	<b>PA0MemType.</b> Read-write. Reset: 6h. MemType for {PAT, PCD, PWT} = 0h.																
	<b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																

**MSR0000\_02FF [MTRR Default Memory Type] (Core::X86::Msrr::MTRRdefType)**

See Core::X86::Msrr::MtrrVarBase for general MTRR information.

\_ccd[1:0]\_lthree0\_core[7:0]; MSR0000\_02FF

Bits	Description
63:12	Reserved.
11	<b>MtrrDefTypeEn: variable and fixed MTRR enable.</b> Read-write. Reset: 0. 0=Fixed and variable MTRRs are not enabled. 1=Core::X86::Msrr::MtrrVarBase, and Core::X86::Msrr::MtrrFix_64K through Core::X86::Msrr::MtrrFix_4K_7 are enabled.
10	<b>MtrrDefTypeFixEn: fixed MTRR enable.</b> Read-write. Reset: 0. 0=Core::X86::Msrr::MtrrFix_64K through Core::X86::Msrr::MtrrFix_4K_7 are not enabled. 1=Core::X86::Msrr::MtrrFix_64K through Core::X86::Msrr::MtrrFix_4K_7 are enabled. This field is ignored (and the fixed MTRRs are not enabled) if Core::X86::Msrr::MTRRdefType[MtrrDefTypeEn] == 0.
9:8	Reserved.
7:0	<b>MemType: memory type.</b> Read-write. Reset: 00h. <b>Description:</b> If MtrrDefTypeEn == 1 then MemType specifies the memory type for memory space that is not specified by either the fixed or variable range MTRRs. If MtrrDefTypeEn == 0 then the default memory type for all of memory is UC. Valid encodings are {00000b, Core::X86::Msrr::MtrrFix_64K through Core::X86::Msrr::MtrrFix_4K_7[2:0]}. Other Write values cause a GP(0).

**MSR0000\_06A0 [User CET] (Core::X86::Msrr::U\_CET)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_06A0

Bits	Description
63:2	Reserved.
1	<b>WRSHSTKEN.</b> Read-write. Reset: 0. Enables the WRSS instruction in User Mode.
0	<b>SHSTKEN.</b> Read-write. Reset: 0. When Set Shadow stack is enabled in User mode.

**MSR0000\_06A2 [Supervisor CET] (Core::X86::Msrr::S\_CET)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_06A2

Bits	Description
63:2	Reserved.
1	<b>WRSHSTKEN.</b> Read-write. Reset: 0. Enables the WRSS instruction in Supervisor Mode.
0	<b>SHSTKEN.</b> Read-write. Reset: 0. When Set Shadow stack is enabled in Supervisor mode.

**MSR0000\_06A4 [PL0 Shadow Stack Pointer] (Core::X86::Msrr::PL0Ssp)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_06A4

Bits	Description
63:2	<b>UserLinAddress: PL0 user top of SSP.</b> Read-write. Reset: 0000_0000_0000_0000h. UserLinAddress[63:32] must be zero in 32-bit mode.
1:0	Reserved.

**MSR0000\_06A5 [PL1 Shadow Stack Pointer] (Core::X86::Msr::PL1Ssp)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_06A5

Bits	Description
63:2	<b>UserLinAddress: PL1 user top of SSP.</b> Read-write. Reset: 0000_0000_0000_0000h. UserLinAddress[63:32] must be zero in 32-bit mode.
1:0	Reserved.

**MSR0000\_06A6 [PL2 Shadow Stack Pointer] (Core::X86::Msr::PL2Ssp)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_06A6

Bits	Description
63:2	<b>UserLinAddress: PL2 user top of SSP.</b> Read-write. Reset: 0000_0000_0000_0000h. UserLinAddress[63:32] must be zero in 32-bit mode.
1:0	Reserved.

**MSR0000\_06A7 [PL3 Shadow Stack Pointer] (Core::X86::Msr::PL3Ssp)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_06A7

Bits	Description
63:2	<b>UserLinAddress: PL3 user top of SSP.</b> Read-write. Reset: 0000_0000_0000_0000h. UserLinAddress[63:32] must be zero in 32-bit mode.
1:0	Reserved.

**MSR0000\_06A8 [Interrupt SSP Table Address] (Core::X86::Msr::IstSspAddr)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_06A8

Bits	Description
63:0	<b>IntrLinTableAddress.</b> Read-write. Reset: 0000_0000_0000_0000h. Shadow Stack Pointer interrupt table.

**MSR0000\_0802 [APIC ID] (Core::X86::Msr::APIC\_ID)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0802

Bits	Description
63:32	Reserved.
31:0	<b>ApicId[31:0]: APIC ID[31:0].</b> Reset: XXXX_XXXXh. Local x2APIC ID register. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.

**MSR0000\_0803 [APIC Version] (Core::X86::Msr::ApicVersion)**

_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSR0000_0803	
Bits	Description
63:32	Reserved.
31	<b>ExtApicSpace: extended APIC register space present.</b> Reset: 1. 1=Indicates the presence of extended APIC register space starting at Core::X86::Msr::ExtendedApicFeature. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.
30:25	Reserved.
24	<b>DirectedEoiSupport: directed EOI support.</b> Reset: 1. 0=Directed EOI capability not supported. 1=Directed EOI capability supported. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.
23:16	<b>MaxLvtEntry.</b> Reset: XXh. Specifies the number of entries in the local vector table minus one. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.
15:8	Reserved.
7:0	<b>Version.</b> Reset: 10h. Indicates the version number of this APIC implementation. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.

**MSR0000\_0808 [Task Priority] (Core::X86::Msr::TPR)**

_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSR0000_0808	
Bits	Description
63:8	Reserved.
7:0	<b>Priority.</b> Reset: 00h. This field is assigned by software to set a threshold priority at which the core is interrupted. AccessType: X2APICEN ? Read-write, Volatile : Error-on-read,Error-on-write.

**MSR0000\_0809 [Arbitration Priority] (Core::X86::Msr::ArbitrationPriority)**

Reset: 0000_0000_0000_0000h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSR0000_0809	
Bits	Description
63:8	Reserved.
7:0	<b>Priority.</b> Reset: 00h. Indicates the current priority for a pending interrupt, or a task or interrupt being serviced by the core. The priority is used to arbitrate between cores to determine which accepts a lowest-priority interrupt request. AccessType: X2APICEN ? Read-only,Error-on-write, Volatile : Error-on-read,Error-on-write.

**MSR0000\_080A [Processor Priority] (Core::X86::Msr::ProcessorPriority)**

Reset: 0000_0000_0000_0000h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSR0000_080A	
Bits	Description
63:8	Reserved.
7:0	<b>Priority.</b> Reset: 00h. Indicates the core's current priority servicing a task or interrupt, and is used to determine if any pending interrupts should be serviced. It is the higher value of the task priority value and the current highest in-service interrupt. AccessType: X2APICEN ? Read-only,Error-on-write, Volatile : Error-on-read,Error-on-write.

**MSR0000\_080B [End Of Interrupt] (Core::X86::Msr::EOI)**

Reset: 0000_0000_0000_0000h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSR0000_080B	
Bits	Description
63:0	<b>EOI.</b> Reset: 0000_0000_0000_0000h. A Write zero to this field indicates the end of interrupt processing the currently in service interrupt. AccessType: X2APICEN ? Write-0-only,Error-on-read,Error-on-write-1 : Error-on-read,Error-on-write.

**MSR0000\_080D [Logical Destination Register] (Core::X86::Msr::LDR)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_080D

Bits	Description
63:32	Reserved.
31:16	<b>ClusterDestination.</b> Reset: 0000h. Specifies cluster's destination identification. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.
15:0	<b>LogicalDestination.</b> Reset: 0000h. Specifies one of up to sixteen x2APICs within the cluster specified by ClusterDestination. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write. <b>ValidValues:</b>
Bit	Description
[0]	x2APIC 0
[1]	x2APIC 1
[2]	x2APIC 2
[3]	x2APIC 3
[4]	x2APIC 4
[5]	x2APIC 5
[6]	x2APIC 6
[7]	x2APIC 7
[8]	x2APIC 8
[9]	x2APIC 9
[10]	x2APIC 10
[11]	x2APIC 11
[12]	x2APIC 12
[13]	x2APIC 13
[14]	x2APIC 14
[15]	x2APIC 15

**MSR0000\_080F [Spurious Interrupt Vector] (Core::X86::Msr::SVR)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_080F

Bits	Description
63:10	Reserved.
9	<b>FocusDisable.</b> Reset: 0. 1=Disable focus core checking during lowest-priority arbitrated interrupts. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
8	<b>APICSWEn: APIC software enable.</b> Reset: 0. All LVT entry mask bits are set and cannot be cleared. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
7:0	<b>Vector.</b> Reset: FFh. The vector that is sent to the core in the event of a spurious interrupt. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.



**MSR0000\_081[0...7] [In Service Register] (Core::X86::Msr::ISR)**

Reset: 0000\_0000\_0000\_0000h.

Interrupt In Service status bits [255:0] accessible through 8 ISR registers.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nISR0\_aliasMSR; MSR0000\_0810

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nISR1\_aliasMSR; MSR0000\_0811

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nISR2\_aliasMSR; MSR0000\_0812

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nISR3\_aliasMSR; MSR0000\_0813

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nISR4\_aliasMSR; MSR0000\_0814

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nISR5\_aliasMSR; MSR0000\_0815

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nISR6\_aliasMSR; MSR0000\_0816

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nISR7\_aliasMSR; MSR0000\_0817

**Bits Description**

63:32 Reserved.

31:0 **InServiceBits**. Reset: 0000\_0000h. These bits are set when the corresponding interrupt is being serviced by the core.

AccessType: X2APICEN ? Read-only,Error-on-write,Volatile : Error-on-read,Error-on-write.

**MSR0000\_081[8...F] [Trigger Mode Register] (Core::X86::Msr::TMR)**

Reset: 0000\_0000\_0000\_0000h.

Trigger Mode status bits [255:0] accessible through 8 TMR registers.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nTMR0\_aliasMSR; MSR0000\_0818

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nTMR1\_aliasMSR; MSR0000\_0819

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nTMR2\_aliasMSR; MSR0000\_081A

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nTMR3\_aliasMSR; MSR0000\_081B

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nTMR4\_aliasMSR; MSR0000\_081C

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nTMR5\_aliasMSR; MSR0000\_081D

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nTMR6\_aliasMSR; MSR0000\_081E

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nTMR7\_aliasMSR; MSR0000\_081F

**Bits Description**

63:32 Reserved.

31:0 **TriggerModeBits**. Reset: 0000\_0000h. The corresponding trigger mode bit is updated when an interrupt is accepted.

AccessType: X2APICEN ? Read-only,Error-on-write,Volatile : Error-on-read,Error-on-write.

**ValidValues:**

Value	Description
0	Edge-triggered interrupt
1	Level-triggered interrupt

**MSR0000\_082[0...7] [Interrupt Request Register] (Core::X86::Msr::IRR)**

Reset: 0000\_0000\_0000\_0000h.

Interrupt Request status bits [255:0] accessible through 8 IRR registers.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nIRR0\_aliasMSR; MSR0000\_0820

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nIRR1\_aliasMSR; MSR0000\_0821

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nIRR2\_aliasMSR; MSR0000\_0822

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nIRR3\_aliasMSR; MSR0000\_0823

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nIRR4\_aliasMSR; MSR0000\_0824

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nIRR5\_aliasMSR; MSR0000\_0825

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nIRR6\_aliasMSR; MSR0000\_0826

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nIRR7\_aliasMSR; MSR0000\_0827

**Bits Description**

63:32 Reserved.

31:0 **RequestBits**. Reset: 0000\_0000h. The corresponding request bit is set when the an interrupt is accepted by the x2APIC.

AccessType: X2APICEN ? Read-only,Error-on-write,Volatile : Error-on-read,Error-on-write.

**MSR0000\_0828 [Error Status Register] (Core::X86::Msr::ESR)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0828

Bits	Description
63:8	Reserved.
7	<b>IllegalRegAddr: illegal register address.</b> Reset: 0. This bit indicates that an access to a nonexistent register location within this APIC was attempted. Can only be set in xAPIC mode. AccessType: X2APICEN ? Read,Write-0-only,Error-on-write-1,Volatile : Error-on-read,Error-on-write.
6	<b>RcvdIllegalVector: received illegal vector.</b> Reset: 0. This bit indicates that this APIC has received a message with an illegal vector (00h to 0Fh for fixed and lowest priority interrupts). AccessType: X2APICEN ? Read,Write-0-only,Error-on-write-1,Volatile : Error-on-read,Error-on-write.
5	<b>SentIllegalVector.</b> Reset: 0. This bit indicates that this x2APIC attempted to send a message with an illegal vector (00h to 0Fh for fixed and lowest priority interrupts). AccessType: X2APICEN ? Read,Write-0-only,Error-on-write-1,Volatile : Error-on-read,Error-on-write.
4	Reserved.
3	<b>RcvAcceptError: receive accept error.</b> Reset: 0. This bit indicates that a message received by this APIC was not accepted by this or any other x2APIC. AccessType: X2APICEN ? Read,Write-0-only,Error-on-write-1,Volatile : Error-on-read,Error-on-write.
2	<b>SendAcceptError.</b> Reset: 0. This bit indicates that a message sent by this APIC was not accepted by any x2APIC. AccessType: X2APICEN ? Read,Write-0-only,Error-on-write-1,Volatile : Error-on-read,Error-on-write.
1:0	Reserved.

**MSR0000\_0830 [Interrupt Command] (Core::X86::Msr::InterruptCommand)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0830

Bits	Description																		
63:32	<b>DestinationField.</b> Reset: 0000_0000h. The destination encoding used when Core::X86::Msr::InterruptCommand[DestShrthnd] is 00b. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.																		
31:20	Reserved.																		
19:18	<b>DestShrthnd: destination shorthand.</b> Reset: 0h. Provides a quick way to specify a destination for a message. If all including self or all excluding self is used, then destination mode is ignored and physical is automatically used. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>No shorthand (Destination field).</td></tr> <tr> <td>1h</td><td>Self.</td></tr> <tr> <td>2h</td><td>All including self.</td></tr> <tr> <td>3h</td><td>All excluding self. (This sends a message with a destination encoding of all 1s, so if lowest priority is used the message could end up being reflected back to this APIC.)</td></tr> </table>	Value	Description	0h	No shorthand (Destination field).	1h	Self.	2h	All including self.	3h	All excluding self. (This sends a message with a destination encoding of all 1s, so if lowest priority is used the message could end up being reflected back to this APIC.)								
Value	Description																		
0h	No shorthand (Destination field).																		
1h	Self.																		
2h	All including self.																		
3h	All excluding self. (This sends a message with a destination encoding of all 1s, so if lowest priority is used the message could end up being reflected back to this APIC.)																		
17:16	Reserved.																		
15	<b>TM: trigger mode.</b> Reset: 0. 0=Edge triggered. 1=Level triggered. Indicates how this interrupt is triggered. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.																		
14	<b>Level.</b> Reset: 0. 0=Deasserted. 1=Asserted. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.																		
13:12	Reserved.																		
11	<b>DM: destination mode.</b> Reset: 0. 0=Physical. 1=Logical. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.																		
10:8	<b>MsgType.</b> Reset: 0h. The message types are encoded as follows: AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Fixed</td></tr> <tr> <td>1h</td><td>Lowest Priority.</td></tr> <tr> <td>2h</td><td>SMI</td></tr> <tr> <td>3h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>NMI</td></tr> <tr> <td>5h</td><td>INIT</td></tr> <tr> <td>6h</td><td>Startup</td></tr> <tr> <td>7h</td><td>External interrupt.</td></tr> </table>	Value	Description	0h	Fixed	1h	Lowest Priority.	2h	SMI	3h	Reserved.	4h	NMI	5h	INIT	6h	Startup	7h	External interrupt.
Value	Description																		
0h	Fixed																		
1h	Lowest Priority.																		
2h	SMI																		
3h	Reserved.																		
4h	NMI																		
5h	INIT																		
6h	Startup																		
7h	External interrupt.																		
7:0	<b>Vector.</b> Reset: 00h. The vector that is sent for this interrupt source. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.																		

**MSR0000\_0832 [LVT Timer] (Core::X86::Msr::TimerLvtEntry)**

Reset: 0000\_0000\_0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0832

Bits	Description
63:18	Reserved.
17	<b>Mode.</b> Reset: 0. 0=One-shot. 1=Periodic. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
16	<b>Mask.</b> Reset: 1. 0=Not masked. 1=Masked. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.) AccessType: X2APICEN ? Read-only,Volatile : Error-on-read,Error-on-write.
11:8	Reserved.
7:0	<b>Vector.</b> Reset: 00h. Interrupt vector number. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_0833 [LVT Thermal Sensor] (Core::X86::Msr::ThermalLvtEntry)**

Reset: 0000\_0000\_0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0833

Bits	Description
63:17	Reserved.
16	<b>Mask.</b> Reset: 1. 0=Not masked. 1=Masked. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.) AccessType: X2APICEN ? Read-only,Volatile : Error-on-read,Error-on-write.
11	Reserved.
10:8	<b>MsgType: message type.</b> Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table]. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
7:0	<b>Vector.</b> Reset: 00h. Interrupt vector number. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_0834 [LVT Performance Monitor] (Core::X86::Msr::PerformanceCounterLvtEntry)**

Reset: 0000\_0000\_0001\_0000h.

Interrupts for this local vector table are caused by overflows of:

- Core::X86::Msr::PERF\_LEGACY\_CTL0..3(Performance Event Select [3:0]).
- Core::X86::Msr::PERF\_CTL0..5(Performance Event Select [5:0]).

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0834

Bits	Description
63:17	Reserved.
16	<b>Mask.</b> Reset: 1. 0=Not masked. 1=Masked. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Reset: 0. 0=Idle. 1=Send pending. Indicates that the interrupt has not yet been accepted by the core. AccessType: X2APICEN ? Read-only, Volatile : Error-on-read,Error-on-write.
11	Reserved.
10:8	<b>MsgType: message type.</b> Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table]. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
7:0	<b>Vector.</b> Reset: 00h. Interrupt vector number. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_083[5...6] [LVT LINT[1:0]] (Core::X86::Msr::LVTLINT)**

Reset: 0000\_0000\_0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nLVTLINT0\_aliasMSR; MSR0000\_0835

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_nLVTLINT1\_aliasMSR; MSR0000\_0836

Bits	Description
63:17	Reserved.
16	<b>Mask.</b> Reset: 1. 0=Not masked. 1=Masked. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
15	<b>TM: trigger mode.</b> Reset: 0. 0=Edge. 1=Level. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
14	<b>RmtIRR.</b> Reset: 0. If trigger mode is level, remote Core::X86::Msr::IRR is set when the interrupt has begun service. Remote Core::X86::Msr::IRR is cleared when the end of interrupt has occurred. AccessType: X2APICEN ? Read-only, Volatile : Error-on-read,Error-on-write.
13	Reserved.
12	<b>DS: interrupt delivery status.</b> Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.) AccessType: X2APICEN ? Read-only, Volatile : Error-on-read,Error-on-write.
11	Reserved.
10:8	<b>MsgType: message type.</b> Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table]. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
7:0	<b>Vector.</b> Reset: 00h. Interrupt vector number. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_0837 [LVT Error] (Core::X86::Msr::ErrorLvtEntry)**

Reset: 0000\_0000\_0001\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0837

Bits	Description
63:17	Reserved.
16	<b>Mask.</b> Reset: 1. 0=Not masked. 1=Masked. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Reset: 0. 0=Idle. 1=Send pending. (Indicates that the interrupt has not yet been accepted by the core.) AccessType: X2APICEN ? Read-only,Volatile : Error-on-read,Error-on-write.
11	Reserved.
10:8	<b>MsgType: message type.</b> Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table]. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
7:0	<b>Vector.</b> Reset: 00h. Interrupt vector number. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_0838 [Timer Initial Count] (Core::X86::Msr::TimerInitialCount)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0838

Bits	Description
63:32	Reserved.
31:0	<b>Count.</b> Reset: 0000_0000h. The value copied into the current count register when the timer is loaded or reloaded. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_0839 [Timer Current Count] (Core::X86::Msr::TimerCurrentCount)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0839

Bits	Description
63:32	Reserved.
31:0	<b>Count.</b> Reset: 0000_0000h. The current value of the counter. AccessType: X2APICEN ? Read,Error-on-write,Volatile : Error-on-read,Error-on-write.

**MSR0000\_083E [Timer Divide Configuration] (Core::X86::Msr::TimerDivideConfiguration)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_083E

Bits	Description
63:4	Reserved.
3:0	<b>Div[3:0]</b> . Reset: 0h. Div[2] is unused. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
<b>ValidValues:</b>	
Value	Description
0h	Divide by 2.
1h	Divide by 4.
2h	Divide by 8.
3h	Divide by 16.
7h-4h	Reserved.
8h	Divide by 32.
9h	Divide by 64.
Ah	Divide by 128.
Bh	Divide by 1.
Fh-Ch	Reserved.

**MSR0000\_083F [Self IPI] (Core::X86::Msr::SelfIPI)**

Reset: 0000\_0000\_0000\_0000h.

The self IPI register provides a performance optimized path for sending self IPI's. A self IPI is semantically identical to an inter-processor interrupt sent via the ICR, with a Destination Shorthand of Self, Trigger Mode equal to Edge, and a Delivery Mode equal to Fixed.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_083F

Bits	Description
63:8	Reserved.
7:0	<b>Vector</b> . Reset: 00h. Interrupt vector number. AccessType: X2APICEN ? Write-only,Error-on-read : Error-on-read,Error-on-write.

**MSR0000\_0840 [Extended APIC Feature] (Core::X86::Msr::ExtendedApicFeature)**

Reset: 0000\_0000\_0004\_0007h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0840

Bits	Description
63:24	Reserved.
23:16	<b>ExtLvtCount: extended local vector table count</b> . Reset: 04h. This specifies the number of extended LVT registers (Core::X86::Msr::ExtendedInterruptLvtEntries) in the local APIC. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.
15:3	Reserved.
2	<b>ExtApicIdCap: extended APIC ID capable</b> . Reset: 1. 1=The processor is capable of supporting an 8-bit APIC ID, as controlled by Core::X86::Msr::ExtendedApicControl[ExtApicIdEn]. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.
1	<b>SeoiCap: specific end of interrupt capable</b> . Reset: 1. 1=The Core::X86::Msr::SpecificEndOfInterrupt is present. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.
0	<b>IerCap: interrupt enable register capable</b> . Reset: 1. This bit indicates that the Core::X86::Msr::InterruptEnable0 - 7 are present. See 2.1.11.2.1.8 [Interrupt Masking]. AccessType: X2APICEN ? Read-only,Error-on-write : Error-on-read,Error-on-write.

**MSR0000\_0841 [Extended APIC Control] (Core::X86::Msr::ExtendedApicControl)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0841

Bits	Description
63:3	Reserved.
2	<b>ExtApicIdEn: extended APIC ID enable.</b> Reset: 0. 1=Enable 8-bit APIC ID; Core::X86::Msr::APIC_ID[ApicId[31:0]] supports an 8-bit value; an interrupt broadcast in physical destination mode requires that the (IntDest[7:0] = 1111_1111b) (instead of XXXX_1111b); a match in physical destination mode occurs when (IntDest[7:0] == ApicId[7:0]) instead of (IntDest[3:0] == ApicId[3:0]).
	AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
1	<b>SeoiEn.</b> Reset: 0. 1=Enable SEOI generation when a write to Core::X86::Msr::SpecificEndOfInterrupt is received.
	AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
0	<b>IerEn.</b> Reset: 0. 1=Enable writes to the interrupt enable registers.
	AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_0842 [Specific End Of Interrupt] (Core::X86::Msr::SpecificEndOfInterrupt)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSR0000\_0842

Bits	Description
63:8	Reserved.
7:0	<b>EoiVec: end of interrupt vector.</b> Reset: 00h. A write to this field causes an end of interrupt cycle to be performed for the vector specified in this field. The behavior is undefined if no interrupt is pending for the specified interrupt vector.
	AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_0848 [Interrupt Enable 0] (Core::X86::Msr::InterruptEnable0)**

Reset: 0000\_0000\_FFFF\_FFFFh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0\_aliasMSR; MSR0000\_0848

Bits	Description
63:32	Reserved.
31:16	<b>InterruptEnableBits.</b> Reset: FFFFh. The interrupt enable bits can be used to enable each of the 256 interrupts.
	AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
15:0	Reserved.

**MSR0000\_084[9...F] [Interrupt Enable 7..1] (Core::X86::Msr::InterruptEnable71)**

Reset: 0000\_0000\_FFFF\_FFFFh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1\_aliasMSR; MSR0000\_0849

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2\_aliasMSR; MSR0000\_084A

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3\_aliasMSR; MSR0000\_084B

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n4\_aliasMSR; MSR0000\_084C

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n5\_aliasMSR; MSR0000\_084D

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n6\_aliasMSR; MSR0000\_084E

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n7\_aliasMSR; MSR0000\_084F

Bits	Description
63:32	Reserved.
31:0	<b>InterruptEnableBits.</b> Reset: FFFF_FFFFh. The interrupt enable bits can be used to enable each of the 256 interrupts.
	AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.



**MSR0000\_085[0...3] [Extended Interrupt Local Vector Table] (Core::X86::Msr::ExtendedInterruptLvtEntries)**

Reset: 0000_0000_0001_0000h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n0_aliasMSR; MSR0000_0850	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n1_aliasMSR; MSR0000_0851	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n2_aliasMSR; MSR0000_0852	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n3_aliasMSR; MSR0000_0853	
Bits	Description
63:17	Reserved.
16	<b>Mask.</b> Reset: 1. 0=Not masked. 1=Masked. Interrupt Mask. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
15:13	Reserved.
12	<b>DS: interrupt delivery status.</b> Reset: 0. 0=Idle. 1=Send pending. Indicates that the interrupt has not yet been accepted by the core. AccessType: X2APICEN ? Read-write, Volatile : Error-on-read,Error-on-write.
11	Reserved.
10:8	<b>MsgType: message type.</b> Reset: 0h. See 2.1.11.2.1.14 [Generalized Local Vector Table]. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.
7:0	<b>Vector.</b> Reset: 00h. Interrupt vector number. AccessType: X2APICEN ? Read-write : Error-on-read,Error-on-write.

**MSR0000\_0C8D [Monitoring Event Select] (Core::X86::Msr::QM\_EVTSEL)**

_ccd[1:0]_lthree0; MSR0000_0C8D	
Bits	Description
63:44	Reserved.
43:32	<b>RMID.</b> Read-write. Reset: 000h. Resource Monitoring Identifier.
31	<b>ExtendedEvtID.</b> Read-write. Reset: 0. When set, the EventId fields refers to QoS Extended Feature Identifiers.
30:8	Reserved.
7:0	<b>EventId.</b> Read-write. Reset: 00h. Monitored Event ID.

**MSR0000\_0C8E [QOS L3 Counter] (Core::X86::Msr::QM\_CTR)**

Read,Error-on-write. Reset: 8000_0000_0000_0000h.	
_ccd[1:0]_lthree0; MSR0000_0C8E	
Bits	Description
63	<b>Error.</b> Read,Error-on-write. Reset: 1. Unsupported RMID or event type was written to Core::X86::Msr::QM_EVTSEL.
62	<b>Unavailable.</b> Read,Error-on-write. Reset: 0. Data for this RMID is not available or not monitored for this resource or RMID.
61:0	<b>RmData.</b> Read,Error-on-write. Reset: 0000_0000_0000_0000h. Resource Monitored Data.

**MSR0000\_0C8F (Core::X86::Msr::PQR\_ASSOC)**

Reset: 0000_0000_0000_0000h.	
QOS L2 RMID. The behavior of this register is defined in the AMD64 Technology Platform Quality of Service Extensions specification.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSR0000_0C8F	
Bits	Description
63:36	Reserved.
35:32	<b>Clos.</b> Read-write. Reset: 0h. Class of Service.
31:12	Reserved.
11:0	<b>Rmid.</b> Read-write. Reset: 000h. Resource Monitor Identifier.

**MSR0000\_0DA0 [Extended Supervisor State] (Core::X86::Msr::XSS)**

_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSR0000_0DA0	
Bits	Description
63:13	Reserved.
12	<b>CET_S</b> . Read-write. Reset: 0. System Control-flow Enforcement Technology.
11	<b>CET_U</b> . Read-write. Reset: 0. User Control-flow Enforcement Technology.
10:0	Reserved.

**2.1.13.2 MSRs – MSRC000\_xxxx****MSRC000\_0080 [Extended Feature Enable] (Core::X86::Msr::EFER)**

SKINIT Execution: 0000_0000_0000_0000h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSRC000_0080	
Bits	Description
63:22	Reserved.
21	<b>AutomaticIBRSEn: Automatic IBRS Enable</b> . Read-write. Reset: 0. 0=IBRS protection is not enabled unless (SPEC_CTRL[IBRS] == 1). 1=IBRS protection is enabled for any process running at (CPL == 0) or ((ASID == 0) && SEV-SNP).
20	<b>UAIE: Upper Address Ignore Enable</b> . Read-write. Reset: 0. Upper Address Ignore suppresses canonical faults for most data access virtual addresses, which allows software to use the upper bits of a virtual address as tags.
19	Reserved.
18	<b>IntWbinvdEn</b> . Read-write. Reset: 0. Interruptible wbinvd, wbinvd enable.
17:16	Reserved.
15	<b>TCE: translation cache extension enable</b> . Read-write. Reset: 0. 1=Translation cache extension is enabled. PDC entries related to the linear address of the INVLPG instruction are invalidated. If this bit is 0 all PDC entries are invalidated by the INVLPG instruction.
14	<b>FFXSE: fast FXSAVE/FRSTOR enable</b> . Read-write. Reset: 0. 1=Enables the fast FXSAVE/FRSTOR mechanism. A 64-bit operating system may enable the fast FXSAVE/FRSTOR mechanism if (Core::X86::CpuId::FeatureExtIdEdx[FFXSR] == 1). This bit is set once by the operating system and its value is not changed afterwards.
13	<b>LMSLE: long mode segment limit enable</b> . Read-only, Error-on-write-1. Reset: Fixed, 0. 1=Enables the long mode segment limit check mechanism.
12	<b>SVME: secure virtual machine (SVM) enable</b> . Reset: Fixed, 0. 1=SVM features are enabled. AccessType: Core::X86::Msr::VM_CR[SvmeDisable] ? Read-only, Error-on-write-1 : Read-write.
11	<b>NXE: no-execute page enable</b> . Read-write. Reset: 0. 1=The no-execute page protection feature is enabled.
10	<b>LMA: long mode active</b> . Read-only. Reset: 0. 1=Indicates that long mode is active. When writing the EFER register the value of this bit must be preserved. Software must read the EFER register to determine the value of LMA, change any other bits as required and then write the EFER register. An attempt to write a value that differs from the state determined by hardware results in a #GP fault.
9	Reserved.
8	<b>LME: long mode enable</b> . Read-write. Reset: 0. 1=Long mode is enabled.
7:1	Reserved.
0	<b>SYSCALL: system call extension enable</b> . Read-write. Reset: 0. 1=SYSCALL and SYSRET instructions are enabled. This adds the SYSCALL and SYSRET instructions which can be used in flat addressed operating systems as low latency system calls and returns.

**MSRC000\_0081 [SYSCALL Target Address] (Core::X86::Msr::STAR)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

This register holds the target address used by the SYSCALL instruction and the code and stack segment selector bases used by the SYSCALL and SYSRET instructions.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0081

Bits	Description
63:48	<b>SysRetSel.</b> Read-write. Reset: 0000h. SYSRET CS and SS.
47:32	<b>SysCallSel.</b> Read-write. Reset: 0000h. SYSCALL CS and SS.
31:0	<b>Target.</b> Read-write. Reset: 0000_0000h. SYSCALL target address.

**MSRC000\_0082 [Long Mode SYSCALL Target Address] (Core::X86::Msr::STAR64)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0082

Bits	Description
63:0	<b>LSTAR: long mode target address.</b> Read-write. Reset: 0000_0000_0000_0000h. Target address for 64-bit mode calling programs. The address stored in this register must be in canonical form (if not canonical, a #GP fault occurs).

**MSRC000\_0083 [Compatibility Mode SYSCALL Target Address] (Core::X86::Msr::STARCOMPAT)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0083

Bits	Description
63:0	<b>CSTAR: compatibility mode target address.</b> Read-write. Reset: 0000_0000_0000_0000h. Target address for compatibility mode. The address stored in this register must be in canonical form (if not canonical, a #GP fault occurs).

**MSRC000\_0084 [SYSCALL Flag Mask] (Core::X86::Msr::SYSCALL\_FLAG\_MASK)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0084

Bits	Description
63:32	Reserved.
31:0	<b>Mask: SYSCALL flag mask.</b> Read-write. Reset: 0000_0000h. This register holds the EFLAGS mask used by the SYSCALL instruction. 1=Clear the corresponding EFLAGS bit when executing the SYSCALL instruction.

**MSRC000\_00E7 [Read-Only Max Performance Frequency Clock Count] (Core::X86::Msr::MPerfReadOnly)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_00E7

Bits	Description
63:0	<b>MPerfReadOnly: read-only maximum core clocks counter.</b> Reset: 0000_0000_0000_0000h. Incremented by hardware at the P0 frequency while the core is in C0. In combination with Core::X86::Msr::APerfReadOnly, this is used to determine the effective frequency of the core. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. This field uses software P-state numbering. See Core::X86::Msr::HWCR[EffFreqCntMwait], 2.1.4 [Effective Frequency]. This register is not affected by writes to Core::X86::Msr::MPERF. AccessType: Core::X86::Msr::HWCR[EffFreqReadOnlyLock] ? Read,Error-on-write,Volatile : Read-write,Volatile.

**MSRC000\_00E8 [Read-Only Actual Performance Frequency Clock Count] (Core::X86::Msr::APerfReadOnly)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_00E8

Bits	Description
63:0	<b>APerfReadOnly: read-only actual core clocks counter.</b> Reset: 0000_0000_0000_0000h. This register increments in proportion to the actual number of core clocks cycles while the core is in C0. See Core::X86::Msr::MPerfReadOnly. This register is not affected by writes to Core::X86::Msr::APERF. AccessType: Core::X86::Msr::HWCR[EffFreqReadOnlyLock] ? Read,Error-on-write,Volatile : Read-write,Volatile.

**MSRC000\_00E9 [Instructions Retired Performance Count] (Core::X86::Msr::IRPerfCount)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_00E9

Bits	Description
63:48	Reserved.
47:0	<b>IRPerfCount: instructions retired counter.</b> Reset: 0000_0000_0000h. Dedicated Instructions Retired register increments on once for every instruction retired. See Core::X86::Msr::HWCR[IRPerfEn]. AccessType: Core::X86::Msr::HWCR[EffFreqReadOnlyLock] ? Read,Error-on-write,Volatile : Read-write,Volatile.

**MSRC000\_0100 [FS Base] (Core::X86::Msr::FS\_BASE)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0100

Bits	Description
63:0	<b>FSBase: expanded FS segment base.</b> Read-write. Reset: 0000_0000_0000_0000h. This register provides access to the expanded 64-bit FS segment base. The address stored in this register must be in canonical form (if not canonical, a #GP fault fill occurs).

**MSRC000\_0101 [GS Base] (Core::X86::Msr::GS\_BASE)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0101

Bits	Description
63:0	<b>GSBase: expanded GS segment base.</b> Read-write. Reset: 0000_0000_0000_0000h. This register provides access to the expanded 64-bit GS segment base. The address stored in this register must be in canonical form (if not canonical, a #GP fault fill occurs).

**MSRC000\_0102 [Kernel GS Base] (Core::X86::Msr::KernelGSbase)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0102

Bits	Description
63:0	<b>KernelGSBase: kernel data structure pointer.</b> Read-write. Reset: 0000_0000_0000_0000h. This register holds the kernel data structure pointer which can be swapped with the GS_BASE register using the SwapGS instruction. The address stored in this register must be in canonical form (if not canonical, a #GP fault occurs).

**MSRC000\_0103 [Auxiliary Time Stamp Counter] (Core::X86::Msr::TSC\_AUX)**

Read-write,Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0103

Bits	Description
63:32	Reserved.
31:0	<b>TscAux: auxiliary time stamp counter data.</b> Read-write,Volatile. Reset: 0000_0000h. It is expected that this is initialized by privileged software to a meaningful value, such as a processor ID. This value is returned in the RDTSCP instruction.

**MSRC000\_0104 [Time Stamp Counter Ratio] (Core::X86::Msr::TscRateMsr)**

Core::X86::Msr::TscRateMsr allows the hypervisor to control the guest's view of the Time Stamp Counter. It provides a multiplier that scales the value returned when Core::X86::Msr::TSC[TSC], Core::X86::Msr::MPERF[MPERF], and Core::X86::Msr::MPerfReadOnly[MPerfReadOnly] are read by a guest running under virtualization. This allows the hypervisor to provide a consistent TSC, MPERF, and MPerfReadOnly rate for a guest process when moving that process between cores that have a differing P0 rate. The TSC Ratio MSR does not affect the value read from the TSC, MPERF, and MPerfReadOnly MSRs when read when in host mode or when virtualization is not being used or when accessed by code executed in system management mode (SMM) unless the SMM code is executed within a guest container. The TSC Ratio value does not affect the rate of the underlying TSC, MPERF, and MPerfReadOnly counters, or the value that gets written to the TSC, MPERF, and MPerfReadOnly MSRs counters on a Write by either the host or the guest. The TSC Ratio MSR contains a fixed-point number in 8.32 format, which is 8 bits of integer and 32 bits of fraction. This number is the ratio of the desired P0 frequency to the P0 frequency of the core. The reset value of the TSC Ratio MSR is 1.0, which results when a guest frequency matches the core P0 frequency.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0104

Bits	Description
63:40	Reserved.
39:32	<b>TscRateMsrInt: time stamp counter rate integer.</b> Read-write. Reset: 01h. Specifies the integer part of the MSR TSC ratio value.
31:0	<b>TscRateMsrFrac: time stamp counter rate fraction.</b> Read-write. Reset: 0000_0000h. Specifies the fractional part of the MSR TSC ratio value.

**MSRC000\_0108 [Prefetch Control] (Core::X86::Msr::PrefetchControl)**

Reset: 0000\_0000\_0000\_03C0h.

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC000\_0108

Bits	Description														
63:10	Reserved.														
9:7	<b>PrefetchAggressivenessProfile.</b> Read-write. Reset: 7h. When MasterEnable is set, selects a prefetch aggressiveness profile. <b>ValidValues:</b>														
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Level 0, least aggressive prefetch profile.</td></tr> <tr> <td>1h</td><td>Level 1</td></tr> <tr> <td>2h</td><td>Level 2</td></tr> <tr> <td>3h</td><td>Level 3, most aggressive prefetch profile.</td></tr> <tr> <td>6h-4h</td><td>Reserved.</td></tr> <tr> <td>7h</td><td>Default used by hardware. Not software accessible.</td></tr> </table>	Value	Description	0h	Level 0, least aggressive prefetch profile.	1h	Level 1	2h	Level 2	3h	Level 3, most aggressive prefetch profile.	6h-4h	Reserved.	7h	Default used by hardware. Not software accessible.
Value	Description														
0h	Level 0, least aggressive prefetch profile.														
1h	Level 1														
2h	Level 2														
3h	Level 3, most aggressive prefetch profile.														
6h-4h	Reserved.														
7h	Default used by hardware. Not software accessible.														
6	<b>MasterEnable.</b> Read-write. Reset: 1. Enable prefetch aggressiveness profiles.														
5	<b>UpDown.</b> Read-write. Reset: 0. Disable prefetcher that uses memory access history to determine whether to fetch the next or previous line into L2 cache for all memory accesses.														
4	Reserved.														
3	<b>L2Stream.</b> Read-write. Reset: 0. Disable prefetcher that uses history of memory access patterns to fetch additional sequential lines into L2 cache.														
2	<b>L1Region.</b> Read-write. Reset: 0. Disable prefetcher that uses memory access history to fetch additional lines into L1 cache when the data access for a given instruction tends to be followed by a consistent pattern of other accesses within a localized region.														
1	<b>L1Stride.</b> Read-write. Reset: 0. Disable stride prefetcher that uses memory access history of individual instructions to fetch additional lines into L1 cache when each access is a constant distance from the previous.														
0	<b>L1Stream.</b> Read-write. Reset: 0. Disable stream prefetcher that uses history of memory access patterns to fetch additional sequential lines into L1 cache.														

**MSRC000\_010E [Last Branch Stack Select] (Core::X86::Msr::LastBranchStackSelect)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

This MSR allows Last Branch Stack recording to be suppressed based on branch type and privilege level.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_010E

Bits	Description
63:9	Reserved.
8	<b>FarBranch.</b> Read-write. Reset: 0. When set far branches are not recorded.
7	<b>JmpNearRel.</b> Read-write. Reset: 0. When set, near relative jumps, excluding near relative calls, are not recorded.
6	<b>JmpNearInd.</b> Read-write. Reset: 0. When set, near indirect jumps, excluding near indirect calls and near returns, are not recorded.
5	<b>RetNear.</b> Read-write. Reset: 0. When set, near returns are not recorded.
4	<b>CallNearInd.</b> Read-write. Reset: 0. When set, near indirect calls are not recorded.
3	<b>CallNearRel.</b> Read-write. Reset: 0. When set, near relative calls are not recorded.
2	<b>Jcc.</b> Read-write. Reset: 0. When set conditional branches are not recorded.
1	<b>CplGe0.</b> Read-write. Reset: 0. When set, no branches ending in CPL > 0 are recorded.
0	<b>CplEq0.</b> Read-write. Reset: 0. When set, no branches ending in CPL = 0 are recorded.

**MSRC000\_010F [Debug Extension Control] (Core::X86::Msr::DebugExtnCtl)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_010F

Bits	Description
63:7	Reserved.
6	<b>LBRV2EN.</b> Read-write. Reset: 0. When enabled the last 16 branch targets and from addresses are recorded in LBR_TO_V2_[0..15] and LBR_FROM_V2_[0..15]. Core::X86::Msr::DebugExtnCtl[LBRV2EN] is cleared upon #DB entry.
5:0	Reserved.

**MSRC000\_0300 [Performance Counter Global Status] (Core::X86::Msr::PerfCntrGlobalStatus)**

Read-only. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0300

Bits	Description
63:60	Reserved.
59	<b>PmcFreeze: Performance Counter Freeze.</b> Read-only. Reset: 0. When set, the performance monitor counters have been frozen due an interrupt signaled for the overflow of at least one counter. This feature is enabled via Core::X86::Msr::DBG_CTL_MSR[FPMCI].
58	<b>LbrFreeze: LBR Stack Freeze.</b> Read-only. Reset: 0. When set, LBR stack has been frozen due to an interrupt signaled for the overflow of at least one counter. This feature is enabled via Core::X86::Msr::DBG_CTL_MSR[FLBRI]. LBR Freeze does not affect the legacy LBR registers.
57:6	Reserved.
5:0	<b>PerfCntrOvfl: Performance Counter Overflow Bits.</b> Read-only. Reset: 00h. For each available core Performance Counter there is one overflow bit starting at bit position 0 for counter 0 (PerfCntr0). The bit is set when the corresponding counter overflows and remains set until cleared by software via Core::X86::Msr::PerfCntrGlobalStatusClr. The bits can also be set directly by software via Core::X86::Msr::PerfCntrGlobalStatusSet. Note that the overflow bit is set even when the Performance Counter was not configured to signal an interrupt.



**MSRC000\_0301 [Performance Counter Global Control] (Core::X86::Msr::PerfCntrGlobalCtl)**

Read-write. Reset: 0000\_0000\_0000\_003Fh.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0301

Bits	Description
63:6	Reserved.
5:0	<b>PerfCntrEn: Global Performance Counter Enable.</b> Read-write. Reset: 3Fh. For each available performance counter there is one enable bit in this field. Bit position 0 corresponds to counter 0 (PerfCntr0), bit position 1 corresponds to counter 1 (PerfCntr1) and so forth. A Performance counter is enabled to count when both its Core::X86::Msr::PerfCntrGlobalCtl[PerfCntrEn] bit and its Core::X86::Msr::PERF_CTL0..5[En] bit are set.

**MSRC000\_0302 [Performance Counter Global Status Clear] (Core::X86::Msr::PerfCntrGlobalStatusClr)**

Write-only. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0302

Bits	Description
63:60	Reserved.
59	<b>PmcFreezeClr: PMC Freeze Clear.</b> Write-only. Reset: 0. When written as 1, Core::X86::Msr::PerfCntrGlobalStatus[PmcFreeze] is cleared and the Performance Monitor Counter Freeze is cleared and counters that are enabled to count continue to count. Software should use this feature to restart counters after it serviced overflow conditions in a Performance Monitor interrupt handler.
58	<b>LbrFreezeClr: LBR Freeze Clear.</b> Write-only. Reset: 0. When written as 1, Core::X86::Msr::PerfCntrGlobalStatus[LbrFreeze] is cleared, the LBR stack freeze is lifted and continues to record branches as configured. Software should use this feature to restart LBR after it serviced overflow conditions in a Performance Monitor interrupt handler.
57:6	Reserved.
5:0	<b>PerfCntrOvflClr: Performance Counter Overflow Bits Clear.</b> Write-only. Reset: 00h. <b>Description:</b> These bits allow software to clear PerfCntrOvfl bits in Core::X86::Msr::PerfCntrGlobalStatus. To clear a bit software needs to write a 1 to the corresponding bit. Software should clear the Performance Counter Overflow bits when: <ul style="list-style-type: none"> <li>- Handling a performance counter overflow interrupt</li> <li>- Disabling a performance counter</li> <li>- Resetting a performance counter</li> </ul>

**MSRC000\_0303 [Performance Counter Global Status Set] (Core::X86::Msr::PerfCntrGlobalStatusSet)**

Write-only. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC000\_0303

Bits	Description
63:60	Reserved.
59	<b>PmcFreezeSet: PMC Freeze Set.</b> Write-only. Reset: 0. When written as 1, Core::X86::Msr::PerfCntrGlobalStatus[PmcFreeze] is set and the Performance Monitor Counters are frozen. In normal operation it is not expected that software would need to set these bits. It is the hardware that sets this bit when Core::X86::Msr::DBG_CTL_MSR[FPMCI] is set and a PMC overflow signals an interrupt.
58	<b>LbrFreezeClr: LBR Freeze Set.</b> Write-only. Reset: 0. When written as 1, Core::X86::Msr::PerfCntrGlobalStatus[LbrFreeze] is set, the LBR stack is frozen and no longer records branches. In normal operation it is not expected that software would need to set these bits. It is the hardware that sets this bit when Core::X86::Msr::DBG_CTL_MSR[FLBRI] is set and a PMC overflow signals an interrupt.
57:6	Reserved.
5:0	<b>PerfCntrOvflSet: Performance Counter Overflow Bits Set.</b> Write-only. Reset: 00h. These bits allow software to set PerfCntrOvfl bits PerfCntrGlobalStatus. Setting an overflow bit in PerfCntrGlobalStatus does not result in the generation of an interrupt, freeze of performance counters, freeze of LBR or other actions that may be taken when a performance counter overflows.

**MSRC000\_0410 [MCA Interrupt Configuration] (Core::X86::Msr::McaIntrCfg)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

MSRC000\_0410

Bits	Description
63:16	Reserved.
15:12	<b>ThresholdLvtOffset.</b> Read-write. Reset: 0h. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries).
11:8	Reserved.
7:4	<b>DeferredLvtOffset.</b> Read-write. Reset: 0h. For deferred error interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see APIC[530:500]).
3:0	Reserved.



**2.1.13.3 MSRs - MSRC001\_0xxx**

**MSRC001\_0000 [Performance Event Select 0] (Core::X86::Msr::PERF\_LEGACY\_CTL0)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

The legacy alias of Core::X86::Msr::PERF\_CTL0. See Core::X86::Msr::PERF\_CTL0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0000

Bits	Description										
63:42	Reserved.										
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count all events, irrespective of guest/host.</td></tr> <tr> <td>1h</td><td>Count guest events if [SVME] == 1.</td></tr> <tr> <td>2h</td><td>Count host events if [SVME] == 1.</td></tr> <tr> <td>3h</td><td>Count all guest and host events if [SVME] == 1.</td></tr> </table>	Value	Description	0h	Count all events, irrespective of guest/host.	1h	Count guest events if [SVME] == 1.	2h	Count host events if [SVME] == 1.	3h	Count all guest and host events if [SVME] == 1.
Value	Description										
0h	Count all events, irrespective of guest/host.										
1h	Count guest events if [SVME] == 1.										
2h	Count host events if [SVME] == 1.										
3h	Count all guest and host events if [SVME] == 1.										
39:36	Reserved.										
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].										
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.</td></tr> <tr> <td>7Fh-01h</td><td>When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.</td></tr> <tr> <td>FFh-80h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.	7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.	FFh-80h	Reserved.		
Value	Description										
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.										
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.										
FFh-80h	Reserved.										
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.										
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.										
21	Reserved.										
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC Interrupt enable.										
19	Reserved.										
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.										
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. Event counter for OS and user mode.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count no events.</td></tr> <tr> <td>1h</td><td>Count user events (CPL&gt;0).</td></tr> <tr> <td>2h</td><td>Count OS events (CPL=0).</td></tr> <tr> <td>3h</td><td>Count all events, irrespective of the CPL.</td></tr> </table>	Value	Description	0h	Count no events.	1h	Count user events (CPL>0).	2h	Count OS events (CPL=0).	3h	Count all events, irrespective of the CPL.
Value	Description										
0h	Count no events.										
1h	Count user events (CPL>0).										
2h	Count OS events (CPL=0).										
3h	Count all events, irrespective of the CPL.										

15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_0001 [Performance Event Select 1] (Core::X86::Msr::PERF\_LEGACY\_CTL1)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

The legacy alias of Core::X86::Msr::PERF\_CTL1. See Core::X86::Msr::PERF\_CTL1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0001

Bits	Description										
63:42	Reserved.										
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/Guest event counter.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count all events, irrespective of guest/host.</td></tr> <tr> <td>1h</td><td>Count guest events if [SVME] == 1.</td></tr> <tr> <td>2h</td><td>Count host events if [SVME] == 1.</td></tr> <tr> <td>3h</td><td>Count all guest and host events if [SVME] == 1.</td></tr> </table>	Value	Description	0h	Count all events, irrespective of guest/host.	1h	Count guest events if [SVME] == 1.	2h	Count host events if [SVME] == 1.	3h	Count all guest and host events if [SVME] == 1.
Value	Description										
0h	Count all events, irrespective of guest/host.										
1h	Count guest events if [SVME] == 1.										
2h	Count host events if [SVME] == 1.										
3h	Count all guest and host events if [SVME] == 1.										
39:36	Reserved.										
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].										
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.</td></tr> <tr> <td>7Fh-01h</td><td>When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.</td></tr> <tr> <td>FFh-80h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.	7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.	FFh-80h	Reserved.		
Value	Description										
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.										
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.										
FFh-80h	Reserved.										
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.										
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.										
21	Reserved.										
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.										
19	Reserved.										
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.										
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode counter events.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count no events.</td></tr> <tr> <td>1h</td><td>Count user events (CPL&gt;0).</td></tr> <tr> <td>2h</td><td>Count OS events (CPL=0).</td></tr> <tr> <td>3h</td><td>Count all events, irrespective of the CPL.</td></tr> </table>	Value	Description	0h	Count no events.	1h	Count user events (CPL>0).	2h	Count OS events (CPL=0).	3h	Count all events, irrespective of the CPL.
Value	Description										
0h	Count no events.										
1h	Count user events (CPL>0).										
2h	Count OS events (CPL=0).										
3h	Count all events, irrespective of the CPL.										

15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_0002 [Performance Event Select 2] (Core::X86::Msr::PERF\_LEGACY\_CTL2)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

The legacy alias of Core::X86::Msr::PERF\_CTL2. See Core::X86::Msr::PERF\_CTL2.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0002

Bits	Description
63:44	Reserved.
43	<b>PreciseRetire.</b> Read-write. Reset: 0. 0=Include events counted during post-retire speculation that were later aborted. 1=Excludes events counted during post-retire speculation that were later aborted.
42	Reserved.
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/Guest event counter.
<b>ValidValues:</b>	
Value	Description
0h	Count all events, irrespective of guest/host.
1h	Count guest events if [SVME] == 1.
2h	Count host events if [SVME] == 1.
3h	Count all guest and host events if [SVME] == 1.
39:36	Reserved.
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.
<b>ValidValues:</b>	
Value	Description
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.
FFh-80h	Reserved.
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.
21	Reserved.
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.
19	Reserved.
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode.
<b>ValidValues:</b>	
Value	Description
0h	Count no events.
1h	Count user events (CPL>0).
2h	Count OS events (CPL=0).
3h	Count all events, irrespective of the CPL.

15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_0003 [Performance Event Select 3] (Core::X86::Msr::PERF\_LEGACY\_CTL3)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

The legacy alias of Core::X86::Msr::PERF\_CTL3. See Core::X86::Msr::PERF\_CTL3.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0003

Bits	Description										
63:42	Reserved.										
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/guest event counter.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count all events, irrespective of guest/host.</td></tr> <tr> <td>1h</td><td>Count guest events if [SVME] == 1.</td></tr> <tr> <td>2h</td><td>Count host events if [SVME] == 1.</td></tr> <tr> <td>3h</td><td>Count all guest and host events if [SVME] == 1.</td></tr> </table>	Value	Description	0h	Count all events, irrespective of guest/host.	1h	Count guest events if [SVME] == 1.	2h	Count host events if [SVME] == 1.	3h	Count all guest and host events if [SVME] == 1.
Value	Description										
0h	Count all events, irrespective of guest/host.										
1h	Count guest events if [SVME] == 1.										
2h	Count host events if [SVME] == 1.										
3h	Count all guest and host events if [SVME] == 1.										
39:36	Reserved.										
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].										
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.</td></tr> <tr> <td>7Fh-01h</td><td>When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.</td></tr> <tr> <td>FFh-80h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.	7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.	FFh-80h	Reserved.		
Value	Description										
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.										
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.										
FFh-80h	Reserved.										
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.										
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.										
21	Reserved.										
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.										
19	Reserved.										
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.										
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode counter events.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count no events.</td></tr> <tr> <td>1h</td><td>Count user events (CPL&gt;0).</td></tr> <tr> <td>2h</td><td>Count OS events (CPL=0).</td></tr> <tr> <td>3h</td><td>Count all events, irrespective of the CPL.</td></tr> </table>	Value	Description	0h	Count no events.	1h	Count user events (CPL>0).	2h	Count OS events (CPL=0).	3h	Count all events, irrespective of the CPL.
Value	Description										
0h	Count no events.										
1h	Count user events (CPL>0).										
2h	Count OS events (CPL=0).										
3h	Count all events, irrespective of the CPL.										



15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

#### MSRC001\_000[4...7] [Performance Event Counter [3:0]] (Core::X86::Msr::PERF\_LEGACY\_CTR)

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

Note: When counting events that capable of counting greater than 15 events per cycle (MergeEvent) the even and the corresponding odd PERF\_LEGACY\_CTR must be paired to appear as a single 64-bit counter. See 2.1.14.4 [Large Increment per Cycle Events].

The legacy alias of Core::X86::Msr::PERF\_CTR. See Core::X86::Msr::PERF\_CTR.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; MSRC001\_0004

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; MSRC001\_0005

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; MSRC001\_0006

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; MSRC001\_0007

Bits	Description
63:48	Reserved.
47:0	<b>CTR.</b> Read-write, Volatile. Reset: 0000_0000_0000h. Performance counter value.

**MSRC001\_0010 [System Configuration] (Core::X86::Msrr::SYS\_CFG)**

Reset: 0000\_0000\_0000\_0000h.

If Core::X86::Msrr::SYS\_CFG[SecureNestedPagingEn] is set, writes to this register are ignored.

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_0010

Bits	Description															
63:27	Reserved.															
26	<b>HMKEE: Host Multi-Key Encryption Enable.</b> Read,Write-1-only. Reset: 0. Used with SYS_CFG[SMEE] to select secure memory encryption mode. See SYS_CFG[SMEE] for a table listing the available memory encryption modes.															
25	<b>VmplEn.</b> Reset: 0. VM permission levels enable. AccessType: Core::X86::Msrr::SYS_CFG[SecureNestedPagingEn] ? Read-only : Read-write.															
24	<b>SecureNestedPagingEn.</b> Read,Error-on-write-1. Reset: 0. Enable Secure Nested Paging (SNP).															
23	<b>SMEE: Secure Memory Encryption Enable.</b> Read,Write-1-only. Reset: 0. <b>Description:</b> Used with SYS_CFG[HMKEE] to select secure memory encryption mode. See the table below for the available memory encryption modes. <table><tr><th>HMKEE</th><th>SMEE</th><th>Description</th></tr><tr><td>0</td><td>0</td><td>No encryption.</td></tr><tr><td>0</td><td>1</td><td>Enables SME and SEV memory encryption.</td></tr><tr><td>1</td><td>0</td><td>Enables SME-HMK memory encryption.</td></tr><tr><td>1</td><td>1</td><td>Not supported. Results in #GP.</td></tr></table>	HMKEE	SMEE	Description	0	0	No encryption.	0	1	Enables SME and SEV memory encryption.	1	0	Enables SME-HMK memory encryption.	1	1	Not supported. Results in #GP.
HMKEE	SMEE	Description														
0	0	No encryption.														
0	1	Enables SME and SEV memory encryption.														
1	0	Enables SME-HMK memory encryption.														
1	1	Not supported. Results in #GP.														
22	<b>Tom2ForceMemTypeWB: top of memory 2 memory type write back.</b> Read-write. Reset: 0. 1=The default memory type of memory between 4GB and Core::X86::Msrr::TOM2 is write back instead of the memory type defined by Core::X86::Msrr::MTRRdefType[MemType]. For this bit to have any effect, Core::X86::Msrr::MTRRdefType[MtrrDefTypeEn] must be 1. MTRRs and PAT can be used to override this memory type.															
21	<b>MtrrTom2En: MTRR top of memory 2 enable.</b> Read-write. Reset: 0. 0=Core::X86::Msrr::TOM2 is disabled. 1=Core::X86::Msrr::TOM2 is enabled.															
20	<b>MtrrVarDramEn: MTRR variable DRAM enable.</b> Read-write. Reset: 0. Init: BIOS,1. 0=Core::X86::Msrr::TOP_MEM and IORRs are disabled. 1=These registers are enabled.															
19	<b>MtrrFixDramModEn: MTRR fixed RdDram and WrDram modification enable.</b> Read-write. Reset: 0. 0=Core::X86::Msrr::MtrrFix_64K through Core::X86::Msrr::MtrrFix_4K_7 [RdDram,WrDram] read values is masked 00b; writing does not change the hidden value. 1=Core::X86::Msrr::MtrrFix_64K through Core::X86::Msrr::MtrrFix_4K_7 [RdDram,WrDram] access type is Read-write. Not shared between threads. Controls access to Core::X86::Msrr::MtrrFix_64K through Core::X86::Msrr::MtrrFix_4K_7 [RdDram ,WrDram]. This bit should be set to 1 during BIOS initialization of the fixed MTRRs, then cleared to 0 for operation.															
18	<b>MtrrFixDramEn: MTRR fixed RdDram and WrDram attributes enable.</b> Read-write. Reset: 0. Init: BIOS,1. 1=Enables the RdDram and WrDram attributes in Core::X86::Msrr::MtrrFix_64K through Core::X86::Msrr::MtrrFix_4K_7.															
17:0	Reserved.															

**MSRC001\_0015 [Hardware Configuration] (Core::X86::Msr::HWCR)**

Reset: 0000_0000_0100_6010h.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSRC001_0015	
Bits	Description
63:36	Reserved.
35	<b>CpuidFltEn.</b> Read-write. Reset: 0. 1=Executing CPUID outside of SMM and with CPL > 0 results in #GP.
34	<b>DownGradeFp512ToFP256.</b> Read-write. Reset: 0. 1=Downgrade FP512 performance to look more like FP256 performance.
33	<b>SmmPgCfgLock.</b> Read-write. Reset: 0. 1=SMM page config locked. Error-on-write-1 if not in SMM mode. RSM unconditionally clears Core::X86::Msr::HWCR[SmmPgCfgLock].
32:31	Reserved.
30	<b>IRPerfEn: enable instructions retired counter.</b> Read-write. Reset: 0. 1=Enable Core::X86::Msr::IRPerfCount.
29:28	Reserved.
27	<b>EffFreqReadOnlyLock: read-only effective frequency counter lock.</b> Write-1-only. Reset: 0. Init: BIOS, 1. 1=Core::X86::Msr::MPerfReadOnly, Core::X86::Msr::APerfReadOnly and Core::X86::Msr::IRPerfCount are read-only.
26	<b>EffFreqCntMwait: effective frequency counting during mwait.</b> Read-write. Reset: 0. 0=The registers do not increment. 1=The registers increment. Specifies whether Core::X86::Msr::MPERF and Core::X86::Msr::APERF increment while the core is in the monitor event pending state. See 2.1.4 [Effective Frequency].
25	<b>CpbDis: core performance boost disable.</b> Read-write. Reset: 0. 0=CPB is requested to be enabled. 1=CPB is disabled. Specifies whether core performance boost is requested to be enabled or disabled. If core performance boost is disabled while a core is in a boosted P-state, the core automatically transitions to the highest performance non-boosted P-state.
24	<b>TscFreqSel: TSC frequency select.</b> Read-only. Reset: 1. 1=The TSC increments at the P0 frequency.
23:22	Reserved.
21	<b>LockTscToCurrentP0: lock the TSC to the current P0 frequency.</b> Read-write. Reset: 0. 0=The TSC will count at the P0 frequency. 1=The TSC frequency is locked to the current P0 frequency at the time this bit is set and remains fixed regardless of future changes to the P0 frequency.
20	<b>IoCfgGpFault: IO-space configuration causes a GP fault.</b> Read-write. Reset: 0. 1=IO-space accesses to configuration space cause a GP fault. The fault is triggered if any part of the IO Read/Write address range is between CF8h and CFFh, inclusive. These faults only result from single IO instructions, not to string and REP IO instructions. This fault takes priority over the IO trap mechanism described by Core::X86::Msr::SMI_ON_IO_TRAP_CTL_STS.
19	Reserved.
18	<b>McStatusWrEn: machine check status write enable.</b> Read-write. Reset: 0. 0=MCA_STATUS registers are readable; writing a non-zero pattern to these registers causes a general protection fault. 1=MCA_STATUS registers are Read-write, including Reserved fields; do not cause general protection faults; such writes update all implemented bits in these registers; All fields of all threshold registers are Read-write when accessed from MSR space, including Locked, except BlkPtr which is always Read-only; McStatusWrEn does not change the access type for the thresholding registers accessed via configuration space. <b>Description:</b> McStatusWrEn can be used to debug machine check exception and interrupt handlers. Independent of the value of this bit, the processor may enforce Write-Ignored behavior on MCA_STATUS registers depending on platform settings. See 3.1 [Machine Check Architecture].
17	<b>Wrap32Dis: 32-bit address wrap disable.</b> Read-write. Reset: 0. 1=Disable 32-bit address wrapping. Software can use Wrap32Dis to access physical memory above 4 Gbytes without switching into 64-bit mode. To do so, software should write a greater-than 4 Gbyte address to Core::X86::Msr::FS_BASE and Core::X86::Msr::GS_BASE. Then it would address $\pm 2$ Gbytes from one of those bases using normal memory reference instructions with a FS or GS override prefix. However, the INVLPG, FST, and SSE store instructions generate 32-bit addresses in legacy mode, regardless of the state of Wrap32Dis.
16:15	Reserved.

14	<b>RsmSpCycDis: RSM special bus cycle disable.</b> Reset: 1. Init: BIOS,1. 0=A link special bus cycle, SMIACK, is generated on a resume from SMI. AccessType: Core::X86::Msrr::HWCR[SmmLock] ? Read-only : Read-write.
13	<b>SmiSpCycDis: SMI special bus cycle disable.</b> Reset: 1. Init: BIOS,1. 0=A link special bus cycle, SMIACK, is generated when an SMI interrupt is taken. AccessType: Core::X86::Msrr::HWCR[SmmLock] ? Read-only : Read-write.
12:11	Reserved.
10	<b>MonMwaitUserEn: MONITOR/MWAIT user mode enable.</b> Read-write. Reset: 0. 0=The MONITOR and MWAIT instructions are supported only in privilege level 0; these instructions in privilege levels 1 to 3 cause a #UD exception. 1=The MONITOR and MWAIT instructions are supported in all privilege levels. The state of this bit is ignored if MonMwaitDis is set.
9	<b>MonMwaitDis: MONITOR and MWAIT disable.</b> Read-write. Reset: 0. 1=The MONITOR, MWAIT, MONITORX, and MWAITX opcodes become invalid. This affects what is reported back through Core::X86::Cpuid::FeatureIdEcxC[Monitor] and Core::X86::Cpuid::FeatureExtIdEcxC[MwaitExtended].
8	<b>IgnneEm: IGNNE port emulation enable.</b> Read-write. Reset: 0. 1=Enable emulation of IGNNE port.
7	<b>AllowFERRonNE: allow FERR on NE.</b> Read-write. Reset: 0. 0=Disable FERR signalling when generating an x87 floating point exception (when CR0[NE] is set). 1=FERR is signaled on any x87 floating point exception, regardless of CR0[NE].
6:5	Reserved.
4	<b>INVDWBINVD: INVD to WBINVD conversion.</b> Read,Error-on-write-0. Reset: 1. 1=Convert INVD to WBINVD. <b>Description:</b> This bit is required to be set for normal operation when any of the following are true: <ul style="list-style-type: none"> <li>• An L2 is shared by multiple threads.</li> <li>• An L3 is shared by multiple cores.</li> <li>• CC6 is enabled.</li> <li>• Probe filter is enabled.</li> </ul>
3	<b>TlbCacheDis: cacheable memory disable.</b> Read-write. Reset: 0. 1=Disable performance improvement that assumes that the PML4, PDP, PDE and PTE entries are in cacheable WB DRAM. <b>Description:</b> Operating systems that maintain page tables in any other memory type must set the TlbCacheDis bit to insure proper operation. Operating system should do a full TLB flush before and after any changes to this bit value. <ul style="list-style-type: none"> <li>• TlbCacheDis does not override the memory type specified by the SMM ASeg and TSeg memory regions controlled by Core::X86::Msrr::SMMAddr Core::X86::Msrr::SMMMMask.</li> </ul>
2:1	Reserved.
0	<b>SmmLock: SMM code lock.</b> Read,Write-1-only. Reset: 0. Init: BIOS,1. 1=SMM code in the ASeg and TSeg range and the SMM registers are Read-only and SMI interrupts are not intercepted in SVM. See 2.1.11.1.10 [Locking SMM].

**MSRC001\_001[6...8] [IO Range Base] (Core::X86::Msr::IORR\_BASE)**

Read-write.

Core::X86::Msr::IORR\_BASE and Core::X86::Msr::IORR\_MASK combine to specify the two sets of base and mask pairs for two IORR ranges. A core access, with address CPUAddr, is determined to be within IORR address range if the following equation is true:

$\text{CPUAddr}[47:12] \& \text{PhyMask}[47:12] == \text{PhyBase}[47:12] \& \text{PhyMask}[47:12]$ .

BIOS can use the IORRs to create an IO hole within a range of addresses that would normally be mapped to DRAM. It can also use the IORRs to re-assert a DRAM destination for a range of addresses that fall within a bigger IO hole that overlays DRAM.

If Core::X86::Msr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_n0; MSRC001\_0016

\_ccd[1:0]\_lthree0\_core[7:0]\_n1; MSRC001\_0018

Bits	Description
63:48	Reserved.
47:12	<b>PhyBase.</b> Read-write. Reset: X_XXXX_XXXXh. Physical base address for IO range.
11:5	Reserved.
4	<b>RdMem: read from memory.</b> Read-write. Reset: X. 0=Read accesses to the range are directed to IO. 1=Read accesses to the range are directed to system memory.
3	<b>WrMem: write to memory.</b> Read-write. Reset: X. 0=Write accesses to the range are directed to IO. 1=Write accesses to the range are directed to system memory.
2:0	Reserved.

**MSRC001\_001[7...9] [IO Range Mask] (Core::X86::Msr::IORR\_MASK)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See Core::X86::Msr::IORR\_BASE.

If Core::X86::Msr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]\_n0; MSRC001\_0017

\_ccd[1:0]\_lthree0\_core[7:0]\_n1; MSRC001\_0019

Bits	Description
63:48	Reserved.
47:12	<b>PhyMask.</b> Read-write. Reset: 0_0000_0000h. Physical address mask for IO range.
11	<b>Valid.</b> Read-write. Reset: 0. 1=The pair of registers that specifies an IORR range is valid.
10:0	Reserved.

**MSRC001\_001A [Top Of Memory] (Core::X86::Msr::TOP\_MEM)**

Read-write.

If Core::X86::Msr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_001A

Bits	Description
63:48	Reserved.
47:23	<b>TOM[47:23]: top of memory.</b> Read-write. Reset: XXX_XXXXh. Specifies the address that divides between MMIO and DRAM. This value is normally placed below 4G. From TOM to 4G is MMIO; below TOM is DRAM. See 2.1.5.3 [System Address Map].
22:0	Reserved.

**MSRC001\_001D [Top Of Memory 2] (Core::X86::Msr::TOM2)**

Read-write.

If Core::X86::Msr::SYS\_CFG[SecureNestedPagingEn] is set to 1, this register will be read-only and attempts to write to this register will result in #GP(0).

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_001D

Bits	Description
63:48	Reserved.
47:23	<b>TOM2[47:23]: second top of memory.</b> Read-write. Reset: XXX_XXXXh. Specifies the address divides between MMIO and DRAM. This value is normally placed above 4G. From 4G to TOM2 - 1 is DRAM; TOM2 and above is MMIO. See 2.1.5.3 [System Address Map]. This register is enabled by Core::X86::Msr::SYS_CFG[MtrrTom2En].
22:0	Reserved.

**MSRC001\_0020 [Patch Loader] (Core::X86::Msr::PATCH\_LOADER)**

Write-only, Error-on-read.

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_0020

Bits	Description
63:0	<b>PatchBase.</b> Write-only, Error-on-read. Reset: XXXX_XXXX_XXXX_XXXXh. Linear address of the Microcode Patch Block. PatchBase[63:32] is ignored when the core is not operating in 64-bit mode.

**MSRC001\_0022 [Machine Check Exception Redirection] (Core::X86::Msr::McExcepRedir)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

This register can be used to redirect machine check exceptions (MCEs) to SMIs or vectored interrupts. If both RedirSmiEn and RedirVecEn are set, then undefined behavior results.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0022

Bits	Description
63:10	Reserved.
9	<b>RedirSmiEn.</b> Read-write. Reset: 0. 1=Redirect MCEs (that are directed to this core) to generate an SMI-trigger IO cycle via Core::X86::Msr::SmiTrigIoCycle. The status is stored in Core::X86::Smm::LocalSmiStatus[MceRedirSts].
8	<b>RedirVecEn.</b> Read-write. Reset: 0. 1=Redirect MCEs (that are directed to this core) to generate a vectored interrupt, using the interrupt vector specified in RedirVector.
7:0	<b>RedirVector.</b> Read-write. Reset: 00h. See RedirVecEn.

**MSRC001\_003[0...5] [Processor Name String] (Core::X86::Msr::ProcNameString)**

Read-write.

These 6 registers hold the CPUID name string in ASCII. The state of these registers are returned by CPUID instructions, Core::X86::Cpuid::ProcNameStr0Eax through Core::X86::Cpuid::ProcNameStr2Edx. BIOS should set these registers to the product name for the processor as provided by AMD. Each register contains a block of 8 ASCII characters; the least byte corresponds to the first ASCII character of the block; the most-significant byte corresponds to the last character of the block. MSRC001\_0030 contains the first block of the name string; MSRC001\_0035 contains the last block of the name string.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; MSRC001\_0030

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; MSRC001\_0031

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; MSRC001\_0032

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; MSRC001\_0033

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n4; MSRC001\_0034

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n5; MSRC001\_0035

Bits	Description
63:56	<b>CpuNameString7</b> . Read-write. Reset: XXh. CPUID name string in ASCII.
55:48	<b>CpuNameString6</b> . Read-write. Reset: XXh. CPUID name string in ASCII.
47:40	<b>CpuNameString5</b> . Read-write. Reset: XXh. CPUID name string in ASCII.
39:32	<b>CpuNameString4</b> . Read-write. Reset: XXh. CPUID name string in ASCII.
31:24	<b>CpuNameString3</b> . Read-write. Reset: XXh. CPUID name string in ASCII.
23:16	<b>CpuNameString2</b> . Read-write. Reset: XXh. CPUID name string in ASCII.
15:8	<b>CpuNameString1</b> . Read-write. Reset: XXh. CPUID name string in ASCII.
7:0	<b>CpuNameString0</b> . Read-write. Reset: XXh. CPUID name string in ASCII.



**MSRC001\_005[0...3] [IO Trap] (Core::X86::Msr::SMI\_ON\_IO\_TRAP)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Core::X86::Msr::SMI\_ON\_IO\_TRAP and Core::X86::Msr::SMI\_ON\_IO\_TRAP\_CTL\_STS provide a mechanism for executing the SMI handler if an access to one of the specified addresses is detected. Access address and access type checking is performed before IO instruction execution. If the access address and access type match one of the specified IO address and access types, then: (1) the IO instruction is not executed; (2) any breakpoint, other than the single-step breakpoint, set on the IO instruction is not taken (the single-step breakpoint is taken after resuming from SMM); and (3) issue the SMI-trigger IO cycle specified by Core::X86::Msr::SmiTrigIoCycle if enabled. The status is stored in Core::X86::Smm::LocalSmiStatus[IoTrapSts].

IO-space configuration accesses are special IO accesses. An IO access is defined as an IO-space configuration access when IO instruction address bits[31:0] are CFCh, CFDh, CFEh, or CFFh when IO-space configuration is enabled (IO::IoCfgAddr[ConfigEn]). The access address for a configuration space access is the current value of IO::IoCfgAddr[BusNo,Device,Function,RegNo]. The access address for an IO access that is not a configuration access is equivalent to the IO instruction address, bits[31:0].

The access address is compared with SmiAddr, and the instruction access type is compared with the enabled access types defined by ConfigSmi, SmiOnRdEn, and SmiOnWrEn. Access address bits[23:0] can be masked with SmiMask. IO and configuration space trapping to SMI applies only to single IO instructions; it does not apply to string and REP IO instructions. The conditional GP fault described by Core::X86::Msr::HWCR[IoCfgGpFault] takes priority over this trap.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; MSRC001\_0050

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; MSRC001\_0051

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; MSRC001\_0052

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; MSRC001\_0053

Bits	Description						
63	<b>SmiOnRdEn: enable SMI on IO Read.</b> Read-write. Reset: 0. 1=Enables SMI generation on a Read access.						
62	<b>SmiOnWrEn: enable SMI on IO Write.</b> Read-write. Reset: 0. 1=Enables SMI generation on a Write access.						
61	<b>ConfigSmi: configuration space SMI.</b> Read-write. Reset: 0. 0=IO access (that is not an IO-space configuration access). 1=Configuration access.						
60:56	Reserved.						
55:32	<b>SmiMask[23:0].</b> Read-write. Reset: 00_0000h. SMI IO trap mask.						
<b>ValidValues:</b>							
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0</td><td>Mask address bit</td></tr> <tr> <td>1</td><td>Do not mask address bit</td></tr> </table>	Value	Description	0	Mask address bit	1	Do not mask address bit
Value	Description						
0	Mask address bit						
1	Do not mask address bit						
31:0	<b>SmiAddr[31:0].</b> Read-write. Reset: 0000_0000h. SMI IO trap address.						

**MSRC001\_0054 [IO Trap Control] (Core::X86::Msr::SMI\_ON\_IO\_TRAP\_CTL\_STS)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0054

Bits	Description
63:16	Reserved.
15	<b>IoTrapEn: IO trap enable.</b> Read-write. Reset: 0. 1=Enable IO and configuration space trapping specified by Core::X86::Msr::SMI_ON_IO_TRAP and Core::X86::Msr::SMI_ON_IO_TRAP_CTL_STS.
14:8	Reserved.
7	<b>SmiEn3.</b> Read-write. Reset: 0. 1=The trap Core::X86::Msr::SMI_ON_IO_TRAP_n[3] is enabled.
6	Reserved.
5	<b>SmiEn2.</b> Read-write. Reset: 0. 1=The trap Core::X86::Msr::SMI_ON_IO_TRAP_n[2] is enabled.
4	Reserved.
3	<b>SmiEn1.</b> Read-write. Reset: 0. 1=The trap Core::X86::Msr::SMI_ON_IO_TRAP_n[1] is enabled.
2	Reserved.
1	<b>SmiEn0.</b> Read-write. Reset: 0. 1=The trap Core::X86::Msr::SMI_ON_IO_TRAP_n[0] is enabled.
0	Reserved.



**MSRC001\_0055 [Reserved.] (Core::X86::Msr::IntPend)**

Read-only. Reset: Fixed, 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_0055

**Bits Description**

63:0 Reserved.

**MSRC001\_0056 [SMI Trigger IO Cycle] (Core::X86::Msr::SmiTrigIoCycle)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.11.1.3 [SMI Sources And Delivery]. This register specifies an IO cycle that may be generated when a local SMI trigger event occurs. If IoCycleEn is set and there is a local SMI trigger event, then the IO cycle generated is a byte Read or Write, based on IoRd, to address IoPortAddress. If the cycle is a Write, then IoData contains the data written. If the cycle is a Read, the value read is discarded. If IoCycleEn is clear and a local SMI trigger event occurs, then undefined behavior results.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0056

**Bits Description**

63:27 Reserved.

26 **IoRd: IO Read.** Read-write. Reset: 0. 0=IO write. 1=IO read.25 **IoCycleEn: IO cycle enable.** Read-write. Reset: 0. 1=The SMI trigger IO cycle is enabled to be generated.

24 Reserved.

23:16 **IoData.** Read-write. Reset: 00h. See 2.1.11.1.3 [SMI Sources And Delivery].15:0 **IoPortAddress.** Read-write. Reset: 0000h. See 2.1.11.1.3 [SMI Sources And Delivery].

**MSRC001\_0058 [MMIO Configuration Base Address] (Core::X86::Msr::MmioCfgBaseAddr)**

See 2.1.6 [Configuration Space] for a description of MMIO configuration space.

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_0058

Bits	Description																																		
63:48	Reserved.																																		
47:20	<b>MmioCfgBaseAddr[47:20]: MMIO configuration base address bits[47:20].</b> Read-write. Reset: XXX_XXXXh. Specifies the base address of the MMIO configuration range.																																		
19:6	Reserved.																																		
5:2	<b>BusRange: bus range identifier.</b> Read-write. Reset: 0h. Specifies the number of buses in the MMIO configuration space range. The size of the MMIO configuration space is 1 MB times the number of buses. <b>Valid Values:</b>																																		
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>1 bus, 1 segment</td></tr> <tr><td>1h</td><td>2 buses, 1 segment</td></tr> <tr><td>2h</td><td>4 buses, 1 segment</td></tr> <tr><td>3h</td><td>8 buses, 1 segment</td></tr> <tr><td>4h</td><td>16 buses, 1 segment</td></tr> <tr><td>5h</td><td>32 buses, 1 segment</td></tr> <tr><td>6h</td><td>64 buses, 1 segment</td></tr> <tr><td>7h</td><td>128 buses, 1 segment</td></tr> <tr><td>8h</td><td>256 buses, 1 segment</td></tr> <tr><td>9h</td><td>2 segments, 256 buses per segment</td></tr> <tr><td>Ah</td><td>4 segments, 256 buses per segment</td></tr> <tr><td>Bh</td><td>8 segments, 256 buses per segment</td></tr> <tr><td>Ch</td><td>16 segments, 256 buses per segment</td></tr> <tr><td>Dh</td><td>32 segments, 256 buses per segment</td></tr> <tr><td>Eh</td><td>64 segments, 256 buses per segment</td></tr> <tr><td>Fh</td><td>128 segments, 256 buses per segment</td></tr> </table>	Value	Description	0h	1 bus, 1 segment	1h	2 buses, 1 segment	2h	4 buses, 1 segment	3h	8 buses, 1 segment	4h	16 buses, 1 segment	5h	32 buses, 1 segment	6h	64 buses, 1 segment	7h	128 buses, 1 segment	8h	256 buses, 1 segment	9h	2 segments, 256 buses per segment	Ah	4 segments, 256 buses per segment	Bh	8 segments, 256 buses per segment	Ch	16 segments, 256 buses per segment	Dh	32 segments, 256 buses per segment	Eh	64 segments, 256 buses per segment	Fh	128 segments, 256 buses per segment
Value	Description																																		
0h	1 bus, 1 segment																																		
1h	2 buses, 1 segment																																		
2h	4 buses, 1 segment																																		
3h	8 buses, 1 segment																																		
4h	16 buses, 1 segment																																		
5h	32 buses, 1 segment																																		
6h	64 buses, 1 segment																																		
7h	128 buses, 1 segment																																		
8h	256 buses, 1 segment																																		
9h	2 segments, 256 buses per segment																																		
Ah	4 segments, 256 buses per segment																																		
Bh	8 segments, 256 buses per segment																																		
Ch	16 segments, 256 buses per segment																																		
Dh	32 segments, 256 buses per segment																																		
Eh	64 segments, 256 buses per segment																																		
Fh	128 segments, 256 buses per segment																																		
1	Reserved.																																		
0	<b>Enable.</b> Read-write. Reset: 0. 1=MMIO configuration space is enabled.																																		

**MSRC001\_0061 [P-state Current Limit] (Core::X86::Msr::PStateCurLim)**

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_0061

Bits	Description
63:7	Reserved.
6:4	<b>PstateMaxVal: P-state maximum value.</b> Read,Error-on-write,Volatile. Reset: XXXb. Specifies the lowest-performance non-boosted P-state (highest non-boosted value) allowed. Attempts to change Core::X86::Msr::PStateCtl[PstateCmd] to a lower-performance P-state (higher value) are clipped to the value of this field.
3	Reserved.
2:0	<b>CurPstateLimit: current P-state limit.</b> Read,Error-on-write,Volatile. Reset: XXXb. Specifies the highest-performance P-state (lowest value) allowed. CurPstateLimit is always bounded by Core::X86::Msr::PStateCurLim[PstateMaxVal]. Attempts to change the CurPstateLimit to a value greater (lower performance) than Core::X86::Msr::PStateCurLim[PstateMaxVal] leaves CurPstateLimit unchanged.

**MSRC001\_0062 [P-state Control] (Core::X86::Msr::PStateCtl)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0062

Bits	Description
63:3	Reserved.
2:0	<b>PstateCmd: P-state change command.</b> Read-write. Reset: XXXb. Cold reset value varies by product; after a warm reset, value initializes to the P-state the core was in prior to the reset. Writes to this field cause the core to change to the indicated non-boosted P-state number, specified by Core::X86::Msr::PStateDef. 0=P0, 1=P1, etc. P-state limits are applied to any P-state requests made through this register. Reads from this field return the last written value, regardless of whether any limits are applied.

**MSRC001\_0063 [P-state Status] (Core::X86::Msr::PStateStat)**

Read,Error-on-write,Volatile.

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_0063

Bits	Description
63:3	Reserved.
2:0	<b>CurPstate: current P-state.</b> Read,Error-on-write,Volatile. Reset: XXXb. This field provides the frequency component of the current non-boosted P-state of the core (regardless of the source of the P-state change, including Core::X86::Msr::PStateCtl[PstateCmd]. 0=P0, 1=P1, etc.). The value of this field is updated when the COF transitions to a new value associated with a P-state.

**MSRC001\_006[4...B] [P-state [7:0]] (Core::X86::Msr::PStateDef)**

Read-write.

Each of these registers specify the frequency and voltage associated with each of the core P-states.

The CpuVid field in these registers is required to be programmed to the same value in all cores of a processor, but are allowed to be different between processors in a multi-processor system. All other fields in these registers are required to be programmed to the same value in each core of the coherent fabric.

\_n0\_aliasMSR; MSRC001\_0064

\_n1\_aliasMSR; MSRC001\_0065

\_n2\_aliasMSR; MSRC001\_0066

\_n3\_aliasMSR; MSRC001\_0067

\_n4\_aliasMSR; MSRC001\_0068

\_n5\_aliasMSR; MSRC001\_0069

\_n6\_aliasMSR; MSRC001\_006A

\_n7\_aliasMSR; MSRC001\_006B

Bits	Description						
63	<b>PstateEn.</b> Read-write. Reset: X. 0=The P-state specified by this MSR is not valid. 1=The P-state specified by this MSR is valid. The purpose of this register is to indicate if the rest of the P-state information in the register is valid after a reset; it controls no hardware.						
62:33	Reserved.						
32	<b>CpuVid[8]: core VID[8].</b> Read-write. Reset: X.						
31:30	<b>IddDiv: current divisor.</b> Read-write. Reset: XXb. See IddValue.						
29:22	<b>IddValue: current value.</b> Read-write. Reset: XXXXXXXXb. After a reset, IddDiv and IddValue combine to specify the expected maximum current dissipation of a single core that is in the P-state corresponding to the MSR number. These values are intended to be used to create ACPI-defined _PSS objects. The values are expressed in amps; they are not intended to convey final product power levels; they may not match the power levels specified in the Power and Thermal Datasheets.						
21:14	<b>CpuVid[7:0]: core VID[7:0].</b> Read-write. Reset: XXXXXXXXb.						
13:12	Reserved.						
11:0	<b>CpuFid[11:0]: core frequency ID.</b> Read-write. Reset: XXXh. Specifies the core frequency multiplier. The core COF is a function of CpuFid and CpuDid, and defined by CoreCOF.						
	<b>ValidValues:</b>						
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00Fh-000h</td><td>Reserved.</td></tr> <tr> <td>FFFh-010h</td><td>&lt;Value&gt;*5</td></tr> </table>	Value	Description	00Fh-000h	Reserved.	FFFh-010h	<Value>*5
Value	Description						
00Fh-000h	Reserved.						
FFFh-010h	<Value>*5						

**MSRC001\_0073 [C-state Base Address] (Core::X86::Msr::CStateBaseAddr)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0073

Bits	Description
63:16	Reserved.
15:0	<b>CstateAddr: C-state address.</b> Read-write. Reset: 0000h. Specifies the IO addresses trapped by the core for C-state entry requests. A value of 0 in this field specifies that the core does not trap any IO addresses for C-state entry. Writing values greater than FFF8h into this field result in undefined behavior. All other values cause the core to trap IO addresses CstateAddr through CstateAddr + 7.

**MSRC001\_0111 [SMM Base Address] (Core::X86::Msrr::SMM\_BASE)**

Reset: 0000\_0000\_0003\_0000h.

This holds the base of the SMM memory region. The value of this register is stored in the save state on entry into SMM (see 2.1.11.1.5 [SMM Save State]) and it is restored on returning from SMM. The 16-bit CS (code segment) selector is loaded with SmmBase[19:4] on entering SMM. SmmBase[3:0] is required to be 0. The SMM base address can be changed in two ways:

- The SMM base address, at offset FF00h in the SMM state save area, may be changed by the SMI handler. The RSM instruction updates SmmBase with the new value.
- Normal WRMSR access to this register.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0111

Bits	Description
63:32	Reserved.
31:0	<b>SmmBase.</b> Reset: 0003_0000h. Base address of the SMM memory region. AccessType: Core::X86::Msrr::HWCR[SmmLock] ? Read-only : Read-write.

**MSRC001\_0112 [SMM TSeg Base Address] (Core::X86::Msrr::SMMAddr)**

Configurable. Reset: 0000\_0000\_0000\_0000h.

See 2.1.11.1 [System Management Mode (SMM)] and 2.1.5.3.1 [Memory Access to the Physical Address Space]. See Core::X86::Msrr::SMMMask for more information about the ASeg and TSeg address ranges.

Each CPU access, directed at CPUAddr, is determined to be in the TSeg range if the following is true:

$\text{CPUAddr}[47:17] \& \text{TSegMask}[47:17] == \text{TSegBase}[47:17] \& \text{TSegMask}[47:17]$ .

For example, if TSeg spans 256 KB and starts at the 1 MB address. The Core::X86::Msrr::SMMAddr[TSegBase[47:17]] would be set to 0010\_0000h and the Core::X86::Msrr::SMMMask[TSegMask[47:17]] to FFFC\_0000h (with zeros filling in for bits[16:0]). This results in a TSeg range from 0010\_0000 to 0013\_FFFFh.

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_0112

Bits	Description
63:48	Reserved.
47:17	<b>TSegBase[47:17]: TSeg address range base.</b> Configurable. Reset: 0000_0000h. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.
16:0	Reserved.

**MSRC001\_0113 [SMM TSeg Mask] (Core::X86::Msrr::SMMMask)**

Configurable. Reset: 0000\_0000\_0000\_0000h.

See 2.1.11.1 [System Management Mode (SMM)].

The ASeg address range is located at a fixed address from A0000h–BFFFFh. The TSeg range is located at a variable base (specified by Core::X86::Msrr::SMMAddr[TSegBase[47:17]]) with a variable size (specified by Core::X86::Msrr::SMMMask[TSegMask[47:17]]). These ranges provide a safe location for SMM code and data that is not readily accessible by non-SMM applications. The SMI handler can be located in one of these two ranges, or it can be located outside these ranges. These ranges must never overlap each other.

This register specifies how accesses to the ASeg and TSeg address ranges are controlled as follows:

- If [A,T]Valid == 1, then:
  - If in SMM, then:
    - If [A,T]Close == 0, then the accesses are directed to DRAM with memory type as specified in [A,T]MTypeDram.
    - If [A,T]Close == 1, then instruction accesses are directed to DRAM with memory type as specified in [A,T]MTypeDram and data accesses are directed at MMIO space and with attributes based on [A,T]MTypeIoWc.
  - If not in SMM, then the accesses are directed at MMIO space with attributes based on [A,T]MTypeIoWc.
- See 2.1.5.3.1.1 [Determining Memory Type].

\_ccd[1:0]\_lthree0\_core[7:0]; MSRC001\_0113

Bits	Description																
63:48	Reserved.																
47:17	<b>TSegMask[47:17]: TSeg address range mask.</b> Configurable. Reset: 0000_0000h. See Core::X86::Msrr::SMMAddr. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.																
16:15	Reserved.																
14:12	<b>TMTypeDram: TSeg address range memory type.</b> Configurable. Reset: 0h. Specifies the memory type for SMM accesses to the TSeg range that are directed to DRAM. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.																
<b>ValidValues:</b>																	
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>UC or uncacheable.</td></tr> <tr> <td>1h</td><td>WC or write combining.</td></tr> <tr> <td>3h-2h</td><td>Reserved.</td></tr> <tr> <td>4h</td><td>WT or write through.</td></tr> <tr> <td>5h</td><td>WP or write protect.</td></tr> <tr> <td>6h</td><td>WB or write back.</td></tr> <tr> <td>7h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	UC or uncacheable.	1h	WC or write combining.	3h-2h	Reserved.	4h	WT or write through.	5h	WP or write protect.	6h	WB or write back.	7h	Reserved.
Value	Description																
0h	UC or uncacheable.																
1h	WC or write combining.																
3h-2h	Reserved.																
4h	WT or write through.																
5h	WP or write protect.																
6h	WB or write back.																
7h	Reserved.																
11	Reserved.																
10:8	<b>AMTypeDram: ASeg Range Memory Type.</b> Configurable. Reset: 0h. Specifies the memory type for SMM accesses to the ASeg range that are directed to DRAM. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.																

Valid Values:	
Value	Description
0h	UC or uncacheable.
1h	WC or write combining.
3h-2h	Reserved.
4h	WT or write through.
5h	WP or write protect.
6h	WB or write back.
7h	Reserved.
7:6	Reserved.
5	<b>TMTypeIoWc: non-SMM TSeg address range memory type.</b> Configurable. Reset: 0. 0=UC (uncacheable). 1=WC (write combining). Specifies the attribute of TSeg accesses that are directed to MMIO space. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.
4	<b>AMTypeIoWc: non-SMM ASeg address range memory type.</b> Configurable. Reset: 0. 0=UC (uncacheable). 1=WC (write combining). Specifies the attribute of ASeg accesses that are directed to MMIO space. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.
3	<b>TClose: send TSeg address range data accesses to MMIO.</b> Configurable. Reset: 0. 1=When in SMM, direct data accesses in the TSeg address range to MMIO space. See AClose. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.
2	<b>AClose: send ASeg address range data accesses to MMIO.</b> Configurable. Reset: 0. 1=When in SMM, direct data accesses in the ASeg address range to MMIO space. [A,T]Close allows the SMI handler to access the MMIO space located in the same address region as the [A,T]Seg. When the SMI handler is finished accessing the MMIO space, it must clear the bit. Failure to do so before resuming from SMM causes the CPU to erroneously read the save state from MMIO space. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.
1	<b>TValid: enable TSeg SMM address range.</b> Configurable. Reset: 0. 1=The TSeg address range SMM enabled. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.
0	<b>AValid: enable ASeg SMM address range.</b> Configurable. Reset: 0. 1=The ASeg address range SMM enabled. AccessType: (Core::X86::Msrr::HWCR[SmmLock]) ? Read-only : Read-write.

#### MSRC001\_0114 [Virtual Machine Control] (Core::X86::Msrr::VM\_CR)

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0114

Bits	Description
63:5	Reserved.
4	<b>SvmeDisable: SVM disable.</b> Configurable. Reset: 0. 0=Core::X86::Msrr::EFER[SVME] is Read-write. 1=Core::X86::Msrr::EFER[SVME] is Read-only, Error-on-write-1. See Lock for the access type of this field. Attempting to set this field when (Core::X86::Msrr::EFER[SVME] == 1) causes a #GP fault, regardless of the state of Lock. See the docAPM2 section titled "Enabling SVM" for software use of this field.
3	<b>Lock: SVM lock.</b> Read-only, Volatile. Reset: 0. 0=SvmeDisable is Read-write. 1=SvmeDisable is Read-only. See Core::X86::Msrr::SvmLockKey[SvmLockKey] for the condition that causes hardware to clear this field.
2	Reserved.
1	<b>InterceptInit: intercept INIT.</b> Read-write, Volatile. Reset: 0. 0=INIT delivered normally. 1=INIT translated into a SX interrupt. This bit controls how INIT is delivered in host mode. This bit is set by hardware when the SKINIT instruction is executed.
0	Reserved.

**MSRC001\_0115 [IGNNE] (Core::X86::Msr::IGNNE)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0115

Bits	Description
63:1	Reserved.
0	<b>IGNNE: current IGNNE state.</b> Read-write. Reset: 0. This bit controls the current state of the processor internal IGNNE signal.

**MSRC001\_0117 [Virtual Machine Host Save Physical Address] (Core::X86::Msr::VM\_HSAVE\_PA)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0117

Bits	Description
63:48	Reserved.
47:12	<b>VM_HSAVE_PA: physical address of host save area.</b> Read-write. Reset: 0_0000_0000h. This register contains the physical address of a 4-KB region where VMRUN saves host state and where vm-exit restores host state from. Writing this register causes a #GP if (FFFF_FFFF_Fh >= VM_HSAVE_PA >= FFFD_0000_0h) or if either the TSEG or ASEG regions overlap with the range defined by this register.
11:0	Reserved.

**MSRC001\_0118 [SVM Lock Key] (Core::X86::Msr::SvmLockKey)**

Read-write. Reset: Fixed,0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0118

Bits	Description
63:0	<b>SvmLockKey: SVM lock key.</b> Read-write. Reset: Fixed,0000_0000_0000_0000h. Writes to this register when (Core::X86::Msr::VM_CR[Lock] == 0) modify SvmLockKey. If ((Core::X86::Msr::VM_CR[Lock] == 1) && (SvmLockKey!=0) && (The write value == The value stored in SvmLockKey)) for a write to this register, then hardware updates Core::X86::Msr::VM_CR[Lock] = 0. Reads of this register always return zero.

**MSRC001\_011B [AVIC Doorbell] (Core::X86::Msr::AvicDoorbell)**

Write-only,Error-on-read. Reset: 0000\_0000\_0000\_0000h.

The ApicId is a physical APIC Id; not valid for logical APIC ID.

Enable: (Core::X86::Cpuid::SvmRevFeatIdEdx[AVIC] == 1).

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_011B

Bits	Description
63:32	Reserved.
31:0	<b>ApicId: APIC ID [31:0].</b> Write-only,Error-on-read. Reset: 0000_0000h. The value written must be a valid physical APID_ID.

**MSRC001\_0135 [Virtual TOM] (Core::X86::Msr::VIRTUAL\_TOM)**

Configurable. Reset: 0000\_0000\_0000\_0000h.

Access of Core::X86::Msr::VIRTUAL\_TOM in hypervisor mode causes #GP.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0135

Bits	Description
63:52	Reserved.
51:21	<b>VIRTUAL_TOM.</b> Configurable. Reset: 0000_0000h. Guest physical addresses below VIRTUAL_TOM are considered private (C=1) when VIRTUAL_TOM is enabled. Access is AccessType: (SEV_FEATURES[VirtualTom] AND SEV_FEATURES[SNPActive]) ? Read-write : Error-on-read, Error-on-write.
20:0	Reserved.



**MSRC001\_0138 [Secure AVIC Control] (Core::X86::Msr::SecureAVIC)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0138

Bits	Description
63:12	<b>GuestApicBackingPagePtr.</b> Read-write. Reset: 0_0000_0000_0000h. Guest APIC Backing Page Pointer, 4K aligned GPA address.
11:2	Reserved.
1	<b>AllowedNmi.</b> Read-write. Reset: 0. Guest allows host to send NMI.
0	<b>SecureAvicEn.</b> Read-write. Reset: 0. Secure AVIC Enable.

**MSRC001\_0140 [OS Visible Work-around Length] (Core::X86::Msr::OSVW\_ID\_Length)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0140

Bits	Description
63:16	Reserved.
15:0	<b>OSVWIdLength: OS visible work-around ID length.</b> Read-write. Reset: 0000h. See the Revision Guide for the definition of this field; see 1.2 [Reference Documents].

**MSRC001\_0141 [OS Visible Work-around Status] (Core::X86::Msr::OSVW\_Status)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0141

Bits	Description
63:0	<b>OsvwStatusBits: OS visible work-around status bits.</b> Read-write. Reset: 0000_0000_0000_0000h. See the Revision Guide for the definition of this field; see 1.2 [Reference Documents].

**MSRC001\_0200 [Performance Event Select 0] (Core::X86::Msr::PERF\_CTL0)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.14 [Performance Monitor Counters]. Core::X86::Msr::PERF\_LEGACY\_CTL0 is an alias of this register.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0200

Bits	Description										
63:42	Reserved.										
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/Guest event counter.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count all events, irrespective of guest/host.</td></tr> <tr> <td>1h</td><td>Count guest events if [SVME] == 1.</td></tr> <tr> <td>2h</td><td>Count host events if [SVME] == 1.</td></tr> <tr> <td>3h</td><td>Count all guest and host events if [SVME] == 1.</td></tr> </table>	Value	Description	0h	Count all events, irrespective of guest/host.	1h	Count guest events if [SVME] == 1.	2h	Count host events if [SVME] == 1.	3h	Count all guest and host events if [SVME] == 1.
Value	Description										
0h	Count all events, irrespective of guest/host.										
1h	Count guest events if [SVME] == 1.										
2h	Count host events if [SVME] == 1.										
3h	Count all guest and host events if [SVME] == 1.										
39:36	Reserved.										
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].										
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.</td></tr> <tr> <td>7Fh-01h</td><td>When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.</td></tr> <tr> <td>FFh-80h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.	7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.	FFh-80h	Reserved.		
Value	Description										
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.										
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.										
FFh-80h	Reserved.										
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.										
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.										
21	Reserved.										
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.										
19	Reserved.										
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.										
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode counter events.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count no events.</td></tr> <tr> <td>1h</td><td>Count user events (CPL&gt;0).</td></tr> <tr> <td>2h</td><td>Count OS events (CPL=0).</td></tr> <tr> <td>3h</td><td>Count all events, irrespective of the CPL.</td></tr> </table>	Value	Description	0h	Count no events.	1h	Count user events (CPL>0).	2h	Count OS events (CPL=0).	3h	Count all events, irrespective of the CPL.
Value	Description										
0h	Count no events.										
1h	Count user events (CPL>0).										
2h	Count OS events (CPL=0).										
3h	Count all events, irrespective of the CPL.										

15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

#### MSRC001\_020[1...B] [Performance Event Counter [5:0]] (Core::X86::Msr::PERF\_CTR)

Note: When counting events that capable of counting greater than 15 events per cycle (MergeEvent) the even and the corresponding odd PERF\_CTR must be paired to appear as a single 64-bit counter. See 2.1.14.4 [Large Increment per Cycle Events].

See Core::X86::Msr::PERF\_CTL0..5. Core::X86::Msr::PERF\_LEGACY\_CTR is an alias of MSRC001\_020[7,5,3,1]. Also can be read via x86 instructions RDPMC ECX=[05:00].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; MSRC001\_0201

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; MSRC001\_0203

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; MSRC001\_0205

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; MSRC001\_0207

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n4; MSRC001\_0209

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n5; MSRC001\_020B

Bits	Description
63:48	Reserved.
47:0	<b>CTR.</b> Read-write, Volatile. Reset: 0000_0000_0000h. Performance counter value.

**MSRC001\_0202 [Performance Event Select 1] (Core::X86::Msr::PERF\_CTL1)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.14 [Performance Monitor Counters]. Core::X86::Msr::PERF\_LEGACY\_CTL1 is an alias of this register.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0202

Bits	Description
63:42	Reserved.
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/Guest event counter.
<b>ValidValues:</b>	
Value	Description
0h	Count all events, irrespective of guest/host.
1h	Count guest events if [SVME] == 1.
2h	Count host events if [SVME] == 1.
3h	Count all guest and host events if [SVME] == 1.
39:36	Reserved.
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.
<b>ValidValues:</b>	
Value	Description
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.
FFh-80h	Reserved.
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.
21	Reserved.
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.
19	Reserved.
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode counter events.
<b>ValidValues:</b>	
Value	Description
0h	Count no events.
1h	Count user events (CPL>0).
2h	Count OS events (CPL=0).
3h	Count all events, irrespective of the CPL.

15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_0204 [Performance Event Select 2] (Core::X86::Msr::PERF\_CTL2)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.14 [Performance Monitor Counters]. Core::X86::Msr::PERF\_LEGACY\_CTL2 is an alias of this register.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0204

Bits	Description
63:44	Reserved.
43	<b>PreciseRetire.</b> Read-write. Reset: 0. 0=Include events counted during post-retire speculation that were later aborted. 1=Excludes events counted during post-retire speculation that were later aborted.
42	Reserved.
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/Guest event counter.
<b>ValidValues:</b>	
Value	Description
0h	Count all events, irrespective of guest/host.
1h	Count guest events if [SVME] == 1.
2h	Count host events if [SVME] == 1.
3h	Count all guest and host events if [SVME] == 1.
39:36	Reserved.
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.
<b>ValidValues:</b>	
Value	Description
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.
FFh-80h	Reserved.
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.
21	Reserved.
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.
19	Reserved.
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode counter events.
<b>ValidValues:</b>	
Value	Description
0h	Count no events.
1h	Count user events (CPL>0).
2h	Count OS events (CPL=0).
3h	Count all events, irrespective of the CPL.

15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_0206 [Performance Event Select 3] (Core::X86::Msr::PERF\_CTL3)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.14 [Performance Monitor Counters]. Core::X86::Msr::PERF\_LEGACY\_CTL3 is an alias of this register.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0206

Bits	Description										
63:42	Reserved.										
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/Guest event counter.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count all events, irrespective of guest/host.</td></tr> <tr> <td>1h</td><td>Count guest events if [SVME] == 1.</td></tr> <tr> <td>2h</td><td>Count host events if [SVME] == 1.</td></tr> <tr> <td>3h</td><td>Count all guest and host events if [SVME] == 1.</td></tr> </table>	Value	Description	0h	Count all events, irrespective of guest/host.	1h	Count guest events if [SVME] == 1.	2h	Count host events if [SVME] == 1.	3h	Count all guest and host events if [SVME] == 1.
Value	Description										
0h	Count all events, irrespective of guest/host.										
1h	Count guest events if [SVME] == 1.										
2h	Count host events if [SVME] == 1.										
3h	Count all guest and host events if [SVME] == 1.										
39:36	Reserved.										
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].										
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.</td></tr> <tr> <td>7Fh-01h</td><td>When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.</td></tr> <tr> <td>FFh-80h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.	7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.	FFh-80h	Reserved.		
Value	Description										
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.										
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.										
FFh-80h	Reserved.										
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.										
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.										
21	Reserved.										
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.										
19	Reserved.										
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.										
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode counter events.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count no events.</td></tr> <tr> <td>1h</td><td>Count user events (CPL&gt;0).</td></tr> <tr> <td>2h</td><td>Count OS events (CPL=0).</td></tr> <tr> <td>3h</td><td>Count all events, irrespective of the CPL.</td></tr> </table>	Value	Description	0h	Count no events.	1h	Count user events (CPL>0).	2h	Count OS events (CPL=0).	3h	Count all events, irrespective of the CPL.
Value	Description										
0h	Count no events.										
1h	Count user events (CPL>0).										
2h	Count OS events (CPL=0).										
3h	Count all events, irrespective of the CPL.										



15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_0208 [Performance Event Select 4] (Core::X86::Msr::PERF\_CTL4)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.14 [Performance Monitor Counters].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_0208

Bits	Description										
63:42	Reserved.										
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/Guest event counters.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count all events, irrespective of guest/host.</td></tr> <tr> <td>1h</td><td>Count guest events if [SVME] == 1.</td></tr> <tr> <td>2h</td><td>Count host events if [SVME] == 1.</td></tr> <tr> <td>3h</td><td>Count all guest and host events if [SVME] == 1.</td></tr> </table>	Value	Description	0h	Count all events, irrespective of guest/host.	1h	Count guest events if [SVME] == 1.	2h	Count host events if [SVME] == 1.	3h	Count all guest and host events if [SVME] == 1.
Value	Description										
0h	Count all events, irrespective of guest/host.										
1h	Count guest events if [SVME] == 1.										
2h	Count host events if [SVME] == 1.										
3h	Count all guest and host events if [SVME] == 1.										
39:36	Reserved.										
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].										
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.</td></tr> <tr> <td>7Fh-01h</td><td>When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.</td></tr> <tr> <td>FFh-80h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.	7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.	FFh-80h	Reserved.		
Value	Description										
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.										
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.										
FFh-80h	Reserved.										
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.										
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.										
21	Reserved.										
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.										
19	Reserved.										
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.										
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode counter events.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count no events.</td></tr> <tr> <td>1h</td><td>Count user events (CPL&gt;0).</td></tr> <tr> <td>2h</td><td>Count OS events (CPL=0).</td></tr> <tr> <td>3h</td><td>Count all events, irrespective of the CPL.</td></tr> </table>	Value	Description	0h	Count no events.	1h	Count user events (CPL>0).	2h	Count OS events (CPL=0).	3h	Count all events, irrespective of the CPL.
Value	Description										
0h	Count no events.										
1h	Count user events (CPL>0).										
2h	Count OS events (CPL=0).										
3h	Count all events, irrespective of the CPL.										

15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_020A [Performance Event Select 5] (Core::X86::Msr::PERF\_CTL5)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.14 [Performance Monitor Counters].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_020A

Bits	Description										
63:42	Reserved.										
41:40	<b>HostGuestOnly: count only host/guest events.</b> Read-write. Reset: 0h. Host/Guest event counter.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count all events, irrespective of guest/host.</td></tr> <tr> <td>1h</td><td>Count guest events if [SVME] == 1.</td></tr> <tr> <td>2h</td><td>Count host events if [SVME] == 1.</td></tr> <tr> <td>3h</td><td>Count all guest and host events if [SVME] == 1.</td></tr> </table>	Value	Description	0h	Count all events, irrespective of guest/host.	1h	Count guest events if [SVME] == 1.	2h	Count host events if [SVME] == 1.	3h	Count all guest and host events if [SVME] == 1.
Value	Description										
0h	Count all events, irrespective of guest/host.										
1h	Count guest events if [SVME] == 1.										
2h	Count host events if [SVME] == 1.										
3h	Count all guest and host events if [SVME] == 1.										
39:36	Reserved.										
35:32	<b>EventSelect[11:8].</b> Read-write. Reset: 0h. Performance event select[11:8].										
31:24	<b>CntMask: counter mask.</b> Read-write. Reset: 00h. Controls the number of events counted per clock cycle.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.</td></tr> <tr> <td>7Fh-01h</td><td>When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.</td></tr> <tr> <td>FFh-80h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.	7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.	FFh-80h	Reserved.		
Value	Description										
00h	The corresponding PERF_CTR[5:0] register increments by the number of events occurring in a clock cycle. See 2.1.14.4 [Large Increment per Cycle Events] for events that can increment greater than 15 per cycle.										
7Fh-01h	When Inv == 0, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is greater than or equal to the CntMask value. When Inv == 1, the corresponding PERF_CTR[5:0] register increments by 1, if the number of events occurring in a clock cycle is less than CntMask value.										
FFh-80h	Reserved.										
23	<b>Inv: invert counter mask.</b> Read-write. Reset: 0. See CntMask.										
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled. Performance counter enable.										
21	Reserved.										
20	<b>Int: enable APIC interrupt.</b> Read-write. Reset: 0. 1=APIC performance counter LVT interrupt is enabled to generate an interrupt via Core::X86::Apic::PerformanceCounterLvtEntry when the performance counter overflows. APIC interrupt enable.										
19	Reserved.										
18	<b>Edge: edge detect.</b> Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. The edge count mode increments the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge is falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.										
17:16	<b>OsUserMode: OS and user mode.</b> Read-write. Reset: 0h. OS and user mode counter events.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Count no events.</td></tr> <tr> <td>1h</td><td>Count user events (CPL&gt;0).</td></tr> <tr> <td>2h</td><td>Count OS events (CPL=0).</td></tr> <tr> <td>3h</td><td>Count all events, irrespective of the CPL.</td></tr> </table>	Value	Description	0h	Count no events.	1h	Count user events (CPL>0).	2h	Count OS events (CPL=0).	3h	Count all events, irrespective of the CPL.
Value	Description										
0h	Count no events.										
1h	Count user events (CPL>0).										
2h	Count OS events (CPL=0).										
3h	Count all events, irrespective of the CPL.										

15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. EventSelect[11:0] = {EventSelect[11:8], EventSelect[7:0]}. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding PERF_CTR[5:0] register. The events are specified in 2.1.14.5 [Core Performance Monitor Counters]. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_023[0...A] [L3 Performance Event Select [5:0]] (Core::X86::Msr::ChL3PmcCfg)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.14.6 [L3 Cache Performance Monitor Counters]

\_ccd[1:0]\_lthree0\_n0; MSRC001\_0230

\_ccd[1:0]\_lthree0\_n1; MSRC001\_0232

\_ccd[1:0]\_lthree0\_n2; MSRC001\_0234

\_ccd[1:0]\_lthree0\_n3; MSRC001\_0236

\_ccd[1:0]\_lthree0\_n4; MSRC001\_0238

\_ccd[1:0]\_lthree0\_n5; MSRC001\_023A

Bits	Description								
63:60	Reserved.								
59:56	<b>ThreadMask.</b> Read-write. Reset: 0h. Controls which of the 2 threads in the selected core are being counted. In non-SMT mode, thread 0 must be selected. One or more threads must be selected unless otherwise specified by the specific L3PMC event. <b>ValidValues:</b> <table> <tr> <th>Bit</th><th>Description</th></tr> <tr> <td>[0]</td><td>Thread 0.</td></tr> <tr> <td>[1]</td><td>Thread 1.</td></tr> <tr> <td>[3:2]</td><td>Reserved.</td></tr> </table>	Bit	Description	[0]	Thread 0.	[1]	Thread 1.	[3:2]	Reserved.
Bit	Description								
[0]	Thread 0.								
[1]	Thread 1.								
[3:2]	Reserved.								
55:51	Reserved.								
50:48	<b>SourceId.</b> Read-write. Reset: 0h. For L3 PMC Events, controls the L3 slice for which events are counted. For Xi PMC Events, controls the CCX interface for which events are counted. Unless otherwise noted by the specific L3PMC event, use Core::X86::Msr::ChL3PmcCfg[SourceId] to select an individual Slice/CCX Interface or Core::X86::Msr::ChL3PmcCfg[EnAllSources] to select all Slices/CCX Interfaces. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>1h-0h</td><td>L3 PMC Event: &lt;Value&gt; Slice, Xi PMC Event: &lt;Value&gt; CCX Interface</td></tr> <tr> <td>7h-2h</td><td>L3 PMC Event: &lt;Value&gt; Slice, Xi PMC Event: Reserved</td></tr> </table>	Value	Description	1h-0h	L3 PMC Event: <Value> Slice, Xi PMC Event: <Value> CCX Interface	7h-2h	L3 PMC Event: <Value> Slice, Xi PMC Event: Reserved		
Value	Description								
1h-0h	L3 PMC Event: <Value> Slice, Xi PMC Event: <Value> CCX Interface								
7h-2h	L3 PMC Event: <Value> Slice, Xi PMC Event: Reserved								
47	<b>EnAllCores.</b> Read-write. Reset: 0. 1=Enable counting L3 events for all cores simultaneously.								
46	<b>EnAllSources.</b> Read-write. Reset: 0. 1=Enable counting. For L3 PMC Events, enable counting on L3 slices simultaneously. For Xi PMC Events, enable counting on all CCX interfaces simultaneously.								
45	Reserved.								
44:42	<b>CoreId.</b> Read-write. Reset: 0h. Controls core for which events are to be counted. See Core::X86::Msr::ChL3PmcCfg[EnAllCores] to count all cores simultaneously. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>7h-0h</td><td>&lt;Value&gt; CoreId.</td></tr> </table>	Value	Description	7h-0h	<Value> CoreId.				
Value	Description								
7h-0h	<Value> CoreId.								
41:23	Reserved.								
22	<b>Enable: Enable L3 performance counter.</b> Read-write. Reset: 0. 1=Enable.								
21:16	Reserved.								
15:8	<b>UnitMask: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored. Unless otherwise stated, the UnitMask values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UnitMask value is an ordinal rather than a bit mask. These situations are described where applicable, or should be obvious from the event descriptions. For events where no UnitMask table is shown, the UnitMask is Unused. When selecting an event for which not all UnitMask bits are defined, the undefined UnitMask bits should be set to zero.								
7:0	<b>EventSel.</b> Read-write. Reset: 00h. L3 Event select.								

**MSRC001\_02[40...5E] [Data Fabric Performance Event Select [3:0]] (Core::X86::Msr::DF\_PERF\_CTL)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

See 2.1.14 [Performance Monitor Counters].

The DF Performance Monitors are shared by all cores/threads in the node. See 2.1.9 [Register Sharing].

\_n0; MSRC001\_0240

\_n1; MSRC001\_0242

\_n2; MSRC001\_0244

\_n3; MSRC001\_0246

\_n4; MSRC001\_0248

\_n5; MSRC001\_024A

\_n6; MSRC001\_024C

\_n7; MSRC001\_024E

\_n8; MSRC001\_0250

\_n9; MSRC001\_0252

\_n10; MSRC001\_0254

\_n11; MSRC001\_0256

\_n12; MSRC001\_0258

\_n13; MSRC001\_025A

\_n14; MSRC001\_025C

\_n15; MSRC001\_025E

Bits	Description
63:39	Reserved.
38:32	<b>EventSelect[14:8]: performance event select.</b> Read-write. Reset: 00h. Performance event select [14:0]. See EventSelect[7:0].
31:28	Reserved.
27:24	<b>UnitMask[11:8]: event qualification.</b> Read-write. Reset: 0h. Uppper 4 bits of UnitMask. See UnitMask[7:0].
23	Reserved.
22	<b>En: enable performance counter.</b> Read-write. Reset: 0. 1=Performance event counter is enabled.
21:16	Reserved.
15:8	<b>UnitMask[7:0]: event qualification.</b> Read-write. Reset: 00h. Each UnitMask bit further specifies or qualifies the event specified by EventSelect. All events selected by UnitMask are simultaneously monitored.
7:0	<b>EventSelect[7:0]: event select.</b> Read-write. Reset: 00h. This field, along with EventSelect[14:8] above, combine to form the 15-bit event select field, EventSelect[14:0]. EventSelect specifies the event or event duration in a processor unit to be counted by the corresponding DF_PERF_CTR[3:0] register. Some events are Reserved; when a Reserved event is selected, the results are undefined.

**MSRC001\_02[41...5F] [Data Fabric Performance Event Counter [3:0]] (Core::X86::Msr::DF\_PERF\_CTR)**

See Core::X86::Msr::DF\_PERF\_CTL. Also can be read via x86 instructions RDPMS ECX=[09:06].  
The DF Performance Monitors are shared by all cores/threads in the socket.

_n0; MSRC001_0241	
_n1; MSRC001_0243	
_n2; MSRC001_0245	
_n3; MSRC001_0247	
_n4; MSRC001_0249	
_n5; MSRC001_024B	
_n6; MSRC001_024D	
_n7; MSRC001_024F	
_n8; MSRC001_0251	
_n9; MSRC001_0253	
_n10; MSRC001_0255	
_n11; MSRC001_0257	
_n12; MSRC001_0259	
_n13; MSRC001_025B	
_n14; MSRC001_025D	
_n15; MSRC001_025F	
Bits	Description
63:48	Reserved.
47:0	<b>CTR[47:0]: performance counter value[47:0].</b> Read-write,Volatile. Reset: 0000_0000_0000h. The current value of the event counter.

**MSRC001\_029A [Core Energy Status] (Core::X86::Msr::CORE\_ENERGY\_STAT)**

Read-only, Volatile. Reset: 0000\_0000\_0000\_0000h.

_ccd[1:0]_lthree0_core[7:0]; MSRC001_029A	
Bits	Description
63:0	<b>TotalEnergyConsumed.</b> Read-only,Volatile. Reset: 0000_0000_0000_0000h.

**MSRC001\_02B0 [CPPC Capability 1] (Core::X86::Msr::CpccCapability1)**

Collaborative Processor Performance Control Capability 1.

_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSRC001_02B0	
Bits	Description
63:32	Reserved.
31:24	<b>HighestPerf: Highest Performance.</b> Read-only,Error-on-write,Volatile. Reset: 00h. Value for the maximum non-ensured performance level.
23:16	<b>NominalPerf: Nominal Performance.</b> Read-only,Error-on-write,Volatile. Reset: 00h. Value for the maximum sustained performance level assuming ideal operating conditions.
15:8	<b>LowNonLinPerf: Lowest Nonlinear Performance.</b> Read-only,Error-on-write,Volatile. Reset: 00h. Value for the lowest nonlinear performance level.
7:0	<b>LowestPerf: Lowest Performance.</b> Read-only,Error-on-write,Volatile. Reset: 00h. Value for the lowest performance level that software can program to MSR_CPPC_REQUEST.

**MSRC001\_02B1 [CPPC Enable] (Core::X86::Msr::CpccEnable)**

Collaborative Processor Performance Control Enable.

MSRC001_02B1	
Bits	Description
63:1	Reserved.
0	<b>CppcEnable.</b> Read,Write-1-only. Reset: 0. CPPC Enable.



**MSRC001\_02B2 [CPPC Capability 2] (Core::X86::Msr::CpccCapability2)**

Collaborative Processor Performance Control Capability 2.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_02B2

Bits	Description
63:8	Reserved.
7:0	<b>MaxPerf: constrained maximum performance.</b> Read-only, Error-on-write, Volatile. Reset: 00h. Value for the current maximum performance level taking into account all known external constraints (i.e., power limits, thermal limits, AC/DC power source, etc.).

**MSRC001\_02B3 [CPPC Request] (Core::X86::Msr::CpccRequest)**

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_02B3

Bits	Description
63:32	Reserved.
31:24	<b>EnergyPerfPref.</b> Read-write. Reset: 00h. Energy Performance Preference.
23:16	<b>DesPerf.</b> Read-write. Reset: 00h. Desired Performance.
15:8	<b>MinPerf.</b> Read-write. Reset: 00h. Minimum Performance.
7:0	<b>MaxPerf.</b> Read-write. Reset: 00h. Maximum Performance.

**MSRC001\_02B4 [CPPC Status] (Core::X86::Msr::CpccStatus)**

Reset: 0000\_0000\_0000\_0000h.

Collaborative Processor Performance Control Status.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_02B4

Bits	Description
63:2	Reserved.
1	<b>MINEXCURSION.</b> Read-write, Volatile. Reset: 0. 1=Minimum Excursion has occurred.
0	Reserved.

**MSRC001\_02F0 [Protected Processor Inventory Number Control] (Core::X86::Msr::PPIN\_CTL)**

Unpredictable.

MSRC001\_02F0

Bits	Description
63:2	Reserved.
1	<b>PPIN_EN.</b> Unpredictable. Reset: X. 0=Reading Core::X86::Msr::PPIN will cause a #GP. 1=Core::X86::Msr::PPIN is accessible using RDMSR. Once set, attempting to write 1 to Core::X86::Msr::PPIN_CTL[Lockout] will cause a #GP.
0	<b>Lockout.</b> Unpredictable. Reset: X. 0=Writes to Core::X86::Msr::PPIN_CTL are permitted if PPIN_EN=0. 1=Further writes to Core::X86::Msr::PPIN_CTL are ignored. <b>Description:</b> Writing 1 to Core::X86::Msr::PPIN_CTL[Lockout] is permitted only if Core::X86::Msr::PPIN_CTL[PPIN_EN] == 0. BIOS should provide an opt-in menu to enable the user to turn on Core::X86::Msr::PPIN_CTL[PPIN_EN] for privileged inventory initialization agent to access Core::X86::Msr::PPIN. After reading Core::X86::Msr::PPIN, the privileged inventory initialization agent should write 00b followed by 01b to Core::X86::Msr::PPIN_CTL to disable further access to MSR_PPIN and prevent unauthorized modification to MSR_PPIN_CTL. Once this bit is written with 1, subsequent writes to this register are ignored, and a reset (warm or cold) is required in order to clear it, which gives BIOS the opportunity to set it again at the next boot.

**MSRC001\_02F1 [Protected Processor Inventory Number] (Core::X86::Msr::PPIN)**

MSRC001\_02F1

Bits	Description
63:0	<b>PPIN.</b> Reset: Fixed,XXXX_XXXX_XXXX_XXXXh. Protected Processor Inventory Number. AccessType: ({Core::X86::Msr::PPIN_CTL[PPIN_EN] , Core::X86::Msr::PPIN_CTL[Lockout]} == 2h) ? Read,Error-on-write : Error-on-read,Error-on-write.

**MSRC001\_03[00...1E] [Access of From IP on Last Branch Record Stack] (Core::X86::Msr::LastBranchStackFromIp)**

Reset: 0000\_0000\_0000\_0000h.

These MSRs capture the branch from IP when LBR V2 is enabled (see Core::X86::Msr::DebugExtnCtl[LBRV2EN]). The youngest branch is in LBR\_FROM\_V2\_00 and the oldest branch is in LBR\_FROM\_V2\_15.

_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n0; MSRC001_0300	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n1; MSRC001_0302	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n2; MSRC001_0304	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n3; MSRC001_0306	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n4; MSRC001_0308	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n5; MSRC001_030A	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n6; MSRC001_030C	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n7; MSRC001_030E	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n8; MSRC001_0310	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n9; MSRC001_0312	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n10; MSRC001_0314	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n11; MSRC001_0316	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n12; MSRC001_0318	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n13; MSRC001_031A	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n14; MSRC001_031C	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_n15; MSRC001_031E	
Bits	Description
63	<b>Mispredict.</b> Read-write. Reset: 0. Indicates if the recorded branch mispredicted.
62:58	<b>BranchFromIpSignExt.</b> Read-write,Volatile. Reset: 00h. Sign extension of BranchFromIp.
57:0	<b>BranchFromIp.</b> Read-write,Volatile. Reset: 000_0000_0000_0000h. Either the lower 63b of the branch segment offset or the segment offset of an instruction preceding the branch. If the segment offset recorded is not the start of the branch instruction, it is the start of a non-branch instruction up to 16 bytes before the end of the branch instruction. The next branch in the sequential path after this instruction is the branch for this LBR Stack entry. Not recording the segment offset of the branch is the result of instruction fusion. If it is desired that BranchFromIp always records the address of branches Instruction Fusion needs to be disabled by also enabling Legacy LBR via Core::X86::Msr::DBG_CTL_MSR[LBR].

**MSRC001\_03[01...1F] [Access of To IP on Last Branch Record Stack] (Core::X86::Msr::LastBranchStackToIp)**

Reset: 0000\_0000\_0000\_0000h.

These MSRs capture the branch to IP and additional information when LBR V2 is enabled (see Core::X86::Msr::DebugExtnCtl[LBRV2EN]). The youngest branch is in LBR\_TO\_V2\_00 and the oldest branch is in LBR\_TO\_V2\_15.

The following table shows the types of branch records based on the Valid and Spec bits:

Valid	Spec	Description
1	0	Normal recorded branch.
1	1	Branch was recorded when speculative performance feature was active.
0	1	Branch was recorded but speculative feature was not successful.
0	0	No branch has yet been logged in this entry since software last cleared this MSR by writing 0.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n0; MSRC001\_0301

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n1; MSRC001\_0303

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n2; MSRC001\_0305

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n3; MSRC001\_0307

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n4; MSRC001\_0309

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n5; MSRC001\_030B

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n6; MSRC001\_030D

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n7; MSRC001\_030F

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n8; MSRC001\_0311

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n9; MSRC001\_0313

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n10; MSRC001\_0315

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n11; MSRC001\_0317

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n12; MSRC001\_0319

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n13; MSRC001\_031B

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n14; MSRC001\_031D

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_n15; MSRC001\_031F

Bits	Description
63	<b>Valid.</b> Read-write. Reset: 0. The valid bit is cleared when the branch was speculatively recorded by the hardware but eventually discarded. Valid LastBranchFrom/To entries have this bit set.
62	<b>Spec: Speculative.</b> Read-write. Reset: 0. When set, the entry was written speculatively .
61	Reserved.
60:58	<b>BranchToIpSignExt.</b> Read-write, Volatile. Reset: 0h. Sign extension of BranchToIp.
57:0	<b>BranchToIp.</b> Read-write, Volatile. Reset: 000_0000_0000_0000h. Lower 58b of the branch target code segment offset.

**MSRC001\_08[00...7E] [UMC Performance Monitor Control] (Core::X86::MsR::UMC\_PerfMonCtl)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

These MSRs provide access to the UMC Performance Monitor Control registers. Refer to Core::X86::Cpuid::ExtPerfMonAndDbgEbx[NumPerfCtrUmc] and Core::X86::Cpuid::ExtPerfMonAndDbgEcx[ActiveUmcMask] for the number of available UMC PMCs and the number of PMCs per UMC. MSRs belonging to non-existent UMC Performance Monitors return zero when read and ignore writes. Reserved bits must be set to 0 for reliable results.

\_n0; MSRC001\_0800

\_n1; MSRC001\_0802

\_n2; MSRC001\_0804

\_n3; MSRC001\_0806

\_n4; MSRC001\_0808

\_n5; MSRC001\_080A

\_n6; MSRC001\_080C

\_n7; MSRC001\_080E

\_n8; MSRC001\_0810

\_n9; MSRC001\_0812

\_n10; MSRC001\_0814

\_n11; MSRC001\_0816

\_n12; MSRC001\_0818

\_n13; MSRC001\_081A

\_n14; MSRC001\_081C

\_n15; MSRC001\_081E

\_n16; MSRC001\_0820

\_n17; MSRC001\_0822

\_n18; MSRC001\_0824

\_n19; MSRC001\_0826

\_n20; MSRC001\_0828

\_n21; MSRC001\_082A

\_n22; MSRC001\_082C

\_n23; MSRC001\_082E

\_n24; MSRC001\_0830

\_n25; MSRC001\_0832

\_n26; MSRC001\_0834

\_n27; MSRC001\_0836

\_n28; MSRC001\_0838

\_n29; MSRC001\_083A

\_n30; MSRC001\_083C

\_n31; MSRC001\_083E

\_n32; MSRC001\_0840

\_n33; MSRC001\_0842

\_n34; MSRC001\_0844

\_n35; MSRC001\_0846

\_n36; MSRC001\_0848

\_n37; MSRC001\_084A

\_n38; MSRC001\_084C

\_n39; MSRC001\_084E

\_n40; MSRC001\_0850

\_n41; MSRC001\_0852

\_n42; MSRC001\_0854

\_n43; MSRC001\_0856

\_n44; MSRC001\_0858

\_n45; MSRC001\_085A

\_n46; MSRC001\_085C

\_n47; MSRC001\_085E

\_n48; MSRC001\_0860

\_n49; MSRC001\_0862

\_n50; MSRC001\_0864

\_n51; MSRC001\_0866

\_n52; MSRC001\_0868

\_n53; MSRC001\_086A

\_n54; MSRC001\_086C

\_n55; MSRC001\_086E

_n56; MSRC001_0870											
_n57; MSRC001_0872											
_n58; MSRC001_0874											
_n59; MSRC001_0876											
_n60; MSRC001_0878											
_n61; MSRC001_087A											
_n62; MSRC001_087C											
_n63; MSRC001_087E											
Bits	Description										
63:32	Reserved.										
31	<b>Enable.</b> Read-write. Reset: 0. 0=Disable. 1=Enable. Counter enable.										
30:10	Reserved.										
9:8	<b>RdWrMask.</b> Read-write. Reset: 0h. Mask transactions based on read or write.										
	<b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>No masking, includes reads and writes.</td></tr> <tr> <td>1h</td><td>Mask writes.</td></tr> <tr> <td>2h</td><td>Mask reads.</td></tr> <tr> <td>3h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	No masking, includes reads and writes.	1h	Mask writes.	2h	Mask reads.	3h	Reserved.
Value	Description										
0h	No masking, includes reads and writes.										
1h	Mask writes.										
2h	Mask reads.										
3h	Reserved.										
7:0	<b>EventSelect.</b> Read-write. Reset: 00h. Select the performance monitor event.										

**MSRC001\_08[01...7F] [UMC Performance Monitor Counter] (Core::X86::Msr::UMC\_PerfMonCntr)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

These MSRs provide access to the UMC Performance Monitor Counter registers. Refer to Core::X86::Cpuid::ExtPerfMonAndDbgEbx[NumPerfCtrUmc] and Core::X86::Cpuid::ExtPerfMonAndDbgEcx[ActiveUmcMask] for the number of available UMC PMCs and the number of PMCs per UMC. MSRs belonging to non-existent UMC Performance Monitors return zero when read and ignore writes.

\_n0; MSRC001\_0801

\_n1; MSRC001\_0803

\_n2; MSRC001\_0805

\_n3; MSRC001\_0807

\_n4; MSRC001\_0809

\_n5; MSRC001\_080B

\_n6; MSRC001\_080D

\_n7; MSRC001\_080F

\_n8; MSRC001\_0811

\_n9; MSRC001\_0813

\_n10; MSRC001\_0815

\_n11; MSRC001\_0817

\_n12; MSRC001\_0819

\_n13; MSRC001\_081B

\_n14; MSRC001\_081D

\_n15; MSRC001\_081F

\_n16; MSRC001\_0821

\_n17; MSRC001\_0823

\_n18; MSRC001\_0825

\_n19; MSRC001\_0827

\_n20; MSRC001\_0829

\_n21; MSRC001\_082B

\_n22; MSRC001\_082D

\_n23; MSRC001\_082F

\_n24; MSRC001\_0831

\_n25; MSRC001\_0833

\_n26; MSRC001\_0835

\_n27; MSRC001\_0837

\_n28; MSRC001\_0839

\_n29; MSRC001\_083B

\_n30; MSRC001\_083D

\_n31; MSRC001\_083F

\_n32; MSRC001\_0841

\_n33; MSRC001\_0843

\_n34; MSRC001\_0845

\_n35; MSRC001\_0847

\_n36; MSRC001\_0849

\_n37; MSRC001\_084B

\_n38; MSRC001\_084D

\_n39; MSRC001\_084F

\_n40; MSRC001\_0851

\_n41; MSRC001\_0853

\_n42; MSRC001\_0855

\_n43; MSRC001\_0857

\_n44; MSRC001\_0859

\_n45; MSRC001\_085B

\_n46; MSRC001\_085D

\_n47; MSRC001\_085F

\_n48; MSRC001\_0861

\_n49; MSRC001\_0863

\_n50; MSRC001\_0865

\_n51; MSRC001\_0867

\_n52; MSRC001\_0869

\_n53; MSRC001\_086B

\_n54; MSRC001\_086D

\_n55; MSRC001\_086F

_n56; MSRC001_0871	
_n57; MSRC001_0873	
_n58; MSRC001_0875	
_n59; MSRC001_0877	
_n60; MSRC001_0879	
_n61; MSRC001_087B	
_n62; MSRC001_087D	
_n63; MSRC001_087F	
Bits	Description
63:49	Reserved.
48	<b>Overflow.</b> Read-write. Reset: 0. Performance Counter Overflow bit. Write-0-to-clear.
47:0	<b>Data.</b> Read-write. Reset: 0000_0000_0000h. Performance Counter Value.

#### 2.1.13.4 MSRs - MSRC001\_1xxx

##### MSRC001\_1003 [Thermal and Power Management CPUID Features] (Core::X86::Msr::CPUID\_PWR\_THERM)

Read-write.

Core::X86::Msr::CPUID\_PWR\_THERM provides control over values read from

Core::X86::CpuId::ThermalPwrMgmtEcX.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1003

Bits	Description
63:1	Reserved.
0	<b>EffFreq.</b> Read-write. Reset: Core::X86::CpuId::ThermalPwrMgmtEcX[EffFreq]. See Core::X86::CpuId::ThermalPwrMgmtEcX[EffFreq].

**MSRC001\_1004 [CPUID Features for CPUID Fn00000001\_E[C,D]X] (Core::X86::Msr::CPUID\_Features)**

Read-write.

Core::X86::Msr::CPUID\_Features[63:32] provides control over values read from Core::X86::Cpuid::FeatureIdEcX; Core::X86::Msr::CPUID\_Features[31:0] provides control over values read from Core::X86::Cpuid::FeatureIdEdX.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1004

Bits	Description
63	Reserved.
62	<b>RDRAND</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[RDRAND]. See Core::X86::Cpuid::FeatureIdEcX[RDRAND].
61	<b>F16C</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[F16C]. See Core::X86::Cpuid::FeatureIdEcX[F16C].
60	<b>AVX</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[AVX]. See Core::X86::Cpuid::FeatureIdEcX[AVX].
59	<b>OSXSAVE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[OSXSAVE]. Modifies Core::X86::Cpuid::FeatureIdEcX[OSXSAVE] only if CR4[OSXSAVE].
58	<b>XSAVE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[XSAVE]. See Core::X86::Cpuid::FeatureIdEcX[XSAVE].
57	<b>AES</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[AES]. Modifies Core::X86::Cpuid::FeatureIdEcX[AES] only if the reset value is 1 .
56	Reserved.
55	<b>POPCNT</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[POPCNT]. See Core::X86::Cpuid::FeatureIdEcX[POPCNT].
54	<b>MOVBE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[MOVBE]. See Core::X86::Cpuid::FeatureIdEcX[MOVBE].
53	<b>X2APIC</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[X2APIC]. See Core::X86::Cpuid::FeatureIdEcX[X2APIC].
52	<b>SSE42</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[SSE42]. See Core::X86::Cpuid::FeatureIdEcX[SSE42].
51	<b>SSE41</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[SSE41]. See Core::X86::Cpuid::FeatureIdEcX[SSE41].
50:46	Reserved.
45	<b>CMPXCHG16B</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[CMXCHG16B]. See Core::X86::Cpuid::FeatureIdEcX[CMXCHG16B].
44	<b>FMA</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[FMA]. See Core::X86::Cpuid::FeatureIdEcX[FMA].
43:42	Reserved.
41	<b>SSSE3</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[SSSE3]. See Core::X86::Cpuid::FeatureIdEcX[SSSE3].
40:36	Reserved.
35	<b>Monitor</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[Monitor]. Modifies Core::X86::Cpuid::FeatureIdEcX[Monitor] only if ~Core::X86::Msr::HWCR[MonMwaitDis].
34	Reserved.
33	<b>PCLMULQDQ</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[PCLMULQDQ]. Modifies Core::X86::Cpuid::FeatureIdEcX[PCLMULQDQ] only if the reset value is 1.
32	<b>SSE3</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEcX[SSE3]. See Core::X86::Cpuid::FeatureIdEcX[SSE3].
31:29	Reserved.
28	<b>HTT</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEdX[HTT]. See Core::X86::Cpuid::FeatureIdEdX[HTT].
27	Reserved.
26	<b>SSE2</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEdX[SSE2]. See Core::X86::Cpuid::FeatureIdEdX[SSE2].
25	<b>SSE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEdX[SSE]. See Core::X86::Cpuid::FeatureIdEdX[SSE].
24	<b>FXSR</b> . Read-write. Reset: Core::X86::Cpuid::FeatureIdEdX[FXSR]. See Core::X86::Cpuid::FeatureIdEdX[FXSR].



23	<b>MMX: MMX instructions.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MMX]. See Core::X86::Cpuid::FeatureIdEdx[MMX].
22:20	Reserved.
19	<b>CLFSH.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[CLFSH]. See Core::X86::Cpuid::FeatureIdEdx[CLFSH].
18	Reserved.
17	<b>PSE36.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PSE36]. See Core::X86::Cpuid::FeatureIdEdx[PSE36].
16	<b>PAT.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PAT]. See Core::X86::Cpuid::FeatureIdEdx[PAT].
15	<b>CMOV.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[CMOV]. See Core::X86::Cpuid::FeatureIdEdx[CMOV].
14	<b>MCA.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MCA]. See Core::X86::Cpuid::FeatureIdEdx[MCA].
13	<b>PGE.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PGE]. See Core::X86::Cpuid::FeatureIdEdx[PGE].
12	<b>MTRR.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MTRR]. See Core::X86::Cpuid::FeatureIdEdx[MTRR].
11	<b>SysEnterSysExit.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[SysEnterSysExit]. See Core::X86::Cpuid::FeatureIdEdx[SysEnterSysExit].
10	Reserved.
9	<b>APIC.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[APIC]. Modifies Core::X86::Cpuid::FeatureIdEdx[APIC] only if Core::X86::Msr::APIC_BAR[ApicEn].
8	<b>CMPXCHG8B.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[CMXCHG8B]. See Core::X86::Cpuid::FeatureIdEdx[CMXCHG8B].
7	<b>MCE.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MCE]. See Core::X86::Cpuid::FeatureIdEdx[MCE].
6	<b>PAE.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PAE]. See Core::X86::Cpuid::FeatureIdEdx[PAE].
5	<b>MSR.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MSR]. See Core::X86::Cpuid::FeatureIdEdx[MSR].
4	<b>TSC.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[TSC]. See Core::X86::Cpuid::FeatureIdEdx[TSC].
3	<b>PSE.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PSE]. See Core::X86::Cpuid::FeatureIdEdx[PSE].
2	<b>DE.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[DE]. See Core::X86::Cpuid::FeatureIdEdx[DE].
1	<b>VME.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[VME]. See Core::X86::Cpuid::FeatureIdEdx[VME].
0	<b>FPU.</b> Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[FPU]. See Core::X86::Cpuid::FeatureIdEdx[FPU].

**MSRC001\_1005 [CPUID Features for CPUID Fn80000001\_E[C,D]X] (Core::X86::Msr::CPUID\_ExtFeatures)**

Read-write.

Core::X86::Msr::CPUID\_ExtFeatures[63:32] provides control over values read from

Core::X86::Cpuid::FeatureExtIdEcX; Core::X86::Msr::CPUID\_ExtFeatures[31:0] provides control over values read from Core::X86::Cpuid::FeatureExtIdEdX.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1005

Bits	Description
63	Reserved.
62	<b>AdMskExtn</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[AdMskExtn]. See Core::X86::Cpuid::FeatureExtIdEcX[AdMskExtn].
61	<b>MwaitExtended</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[MwaitExtended]. See Core::X86::Cpuid::FeatureExtIdEcX[MwaitExtended].
60	<b>PerfCtrExtLLC</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[PerfCtrExtLLC]. See Core::X86::Cpuid::FeatureExtIdEcX[PerfCtrExtLLC].
59	<b>PerfTsc</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[PerfTsc]. See Core::X86::Cpuid::FeatureExtIdEcX[PerfTsc].
58	<b>DataBreakpointExtension</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[DataBreakpointExtension]. See Core::X86::Cpuid::FeatureExtIdEcX[DataBreakpointExtension].
57	Reserved.
56	<b>PerfCtrExtDF</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[PerfCtrExtDF]. See Core::X86::Cpuid::FeatureExtIdEcX[PerfCtrExtDF].
55	<b>PerfCtrExtCore</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[PerfCtrExtCore]. See Core::X86::Cpuid::FeatureExtIdEcX[PerfCtrExtCore].
54	<b>TopologyExtensions</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[TopologyExtensions]. See Core::X86::Cpuid::FeatureExtIdEcX[TopologyExtensions].
53:50	Reserved.
49	<b>TCE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[TCE]. See Core::X86::Cpuid::FeatureExtIdEcX[TCE].
48	<b>FMA4</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[FMA4]. See Core::X86::Cpuid::FeatureExtIdEcX[FMA4].
47	<b>LWP</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[LWP]. See Core::X86::Cpuid::FeatureExtIdEcX[LWP].
46	Reserved.
45	<b>WDT</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[WDT]. See Core::X86::Cpuid::FeatureExtIdEcX[WDT].
44	<b>SKINIT</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[SKINIT]. See Core::X86::Cpuid::FeatureExtIdEcX[SKINIT].
43	<b>XOP</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[XOP]. See Core::X86::Cpuid::FeatureExtIdEcX[XOP].
42	<b>IBS</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[IBS]. See Core::X86::Cpuid::FeatureExtIdEcX[IBS].
41	<b>OSVW</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[OSVW]. See Core::X86::Cpuid::FeatureExtIdEcX[OSVW].
40	<b>ThreeDNowPrefetch</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[ThreeDNowPrefetch]. See Core::X86::Cpuid::FeatureExtIdEcX[ThreeDNowPrefetch].
39	<b>MisAlignSse</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[MisAlignSse]. See Core::X86::Cpuid::FeatureExtIdEcX[MisAlignSse].
38	<b>SSE4A</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcX[SSE4A]. See Core::X86::Cpuid::FeatureExtIdEcX[SSE4A].

37	<b>ABM.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcxC[ABM]. See Core::X86::Cpuid::FeatureExtIdEcxC[ABM].
36	<b>AltMovCr8.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcxC[AltMovCr8]. See Core::X86::Cpuid::FeatureExtIdEcxC[AltMovCr8].
35	<b>ExtApicSpace.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcxC[ExtApicSpace]. See Core::X86::Cpuid::FeatureExtIdEcxC[ExtApicSpace].
34	<b>SVM.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcxC[SVM]. See Core::X86::Cpuid::FeatureExtIdEcxC[SVM].
33	<b>CmpLegacy.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcxC[CmpLegacy]. See Core::X86::Cpuid::FeatureExtIdEcxC[CmpLegacy].
32	<b>LahfSahf.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcxC[LahfSahf]. See Core::X86::Cpuid::FeatureExtIdEcxC[LahfSahf].
31	<b>ThreeDNow: 3DNow! instructions.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[ThreeDNow]. See Core::X86::Cpuid::FeatureExtIdEdxC[ThreeDNow].
30	<b>ThreeDNowExt.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[ThreeDNowExt]. See Core::X86::Cpuid::FeatureExtIdEdxC[ThreeDNowExt].
29	<b>LM.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[LM]. See Core::X86::Cpuid::FeatureExtIdEdxC[LM].
28	Reserved.
27	<b>RDTSCP.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[RDTSCP]. See Core::X86::Cpuid::FeatureExtIdEdxC[RDTSCP].
26	<b>Page1GB.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[Page1GB]. See Core::X86::Cpuid::FeatureExtIdEdxC[Page1GB].
25	<b>FFXSR.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[FFXSR]. See Core::X86::Cpuid::FeatureExtIdEdxC[FFXSR].
24	<b>FXSR.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[FXSR]. See Core::X86::Cpuid::FeatureExtIdEdxC[FXSR].
23	<b>MMX: MMX instructions.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[MMX]. See Core::X86::Cpuid::FeatureExtIdEdxC[MMX].
22	<b>MmxExt.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[MmxExt]. See Core::X86::Cpuid::FeatureExtIdEdxC[MmxExt].
21	Reserved.
20	<b>NX.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[NX]. See Core::X86::Cpuid::FeatureExtIdEdxC[NX].
19:18	Reserved.
17	<b>PSE36.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[PSE36]. See Core::X86::Cpuid::FeatureExtIdEdxC[PSE36].
16	<b>PAT.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[PAT]. See Core::X86::Cpuid::FeatureExtIdEdxC[PAT].
15	<b>CMOV.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[CMOV]. See Core::X86::Cpuid::FeatureExtIdEdxC[CMOV].
14	<b>MCA.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[MCA]. See Core::X86::Cpuid::FeatureExtIdEdxC[MCA].
13	<b>PGE.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[PGE]. See Core::X86::Cpuid::FeatureExtIdEdxC[PGE].
12	<b>MTRR.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[MTRR]. See Core::X86::Cpuid::FeatureExtIdEdxC[MTRR].
11	<b>SysCallSysRet.</b> Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdxC[SysCallSysRet]. See Core::X86::Cpuid::FeatureExtIdEdxC[SysCallSysRet].
10	Reserved.

9	<b>APIC</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[APIC]. See Core::X86::Cpuid::FeatureExtIdEdx[APIC].
8	<b>CMPXCHG8B</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[CMPXCHG8B]. See Core::X86::Cpuid::FeatureExtIdEdx[CMPXCHG8B].
7	<b>MCE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[MCE]. See Core::X86::Cpuid::FeatureExtIdEdx[MCE].
6	<b>PAE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[PAE]. See Core::X86::Cpuid::FeatureExtIdEdx[PAE].
5	<b>MSR</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[MSR]. See Core::X86::Cpuid::FeatureExtIdEdx[MSR].
4	<b>TSC</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[TSC]. See Core::X86::Cpuid::FeatureExtIdEdx[TSC].
3	<b>PSE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[PSE]. See Core::X86::Cpuid::FeatureExtIdEdx[PSE].
2	<b>DE</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[DE]. See Core::X86::Cpuid::FeatureExtIdEdx[DE].
1	<b>VME</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[VME]. See Core::X86::Cpuid::FeatureExtIdEdx[VME].
0	<b>FPU</b> . Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[FPU]. See Core::X86::Cpuid::FeatureExtIdEdx[FPU].

#### MSRC001\_1019 [Address Mask For DR1 Breakpoint] (Core::X86::Msr::DR1\_ADDR\_MASK)

Read-write. Reset: 0000\_0000\_0000\_0000h.

Support indicated by Core::X86::Cpuid::FeatureExtIdEc[DataBreakpointExtension].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1019

Bits	Description
63:32	Reserved.
31:0	<b>AddrMask: mask for DR linear address data breakpoint DR1</b> . Read-write. Reset: 0000_0000h. 1=Exclude bit into address compare. 0=Include bit into address compare. See Core::X86::Msr::DR1_ADDR_MASK. AddrMask[11:0] qualifies the DR1 linear address instruction breakpoint, allowing the DR1 instruction breakpoint on a range of addresses in memory.

#### MSRC001\_101A [Address Mask For DR2 Breakpoint] (Core::X86::Msr::DR2\_ADDR\_MASK)

Read-write. Reset: 0000\_0000\_0000\_0000h.

Support indicated by Core::X86::Cpuid::FeatureExtIdEc[DataBreakpointExtension].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_101A

Bits	Description
63:32	Reserved.
31:0	<b>AddrMask: mask for DR linear address data breakpoint DR2</b> . Read-write. Reset: 0000_0000h. 1=Exclude bit into address compare. 0=Include bit into address compare. See Core::X86::Msr::DR0_ADDR_MASK. AddrMask[11:0] qualifies the DR2 linear address instruction breakpoint, allowing the DR2 instruction breakpoint on a range of addresses in memory.

#### MSRC001\_101B [Address Mask For DR3 Breakpoint] (Core::X86::Msr::DR3\_ADDR\_MASK)

Read-write. Reset: 0000\_0000\_0000\_0000h.

Support indicated by Core::X86::Cpuid::FeatureExtIdEc[DataBreakpointExtension].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_101B

Bits	Description
63:32	Reserved.
31:0	<b>AddrMask: mask for DR linear address data breakpoint DR3</b> . Read-write. Reset: 0000_0000h. 1=Exclude bit into address compare. 0=Include bit into address compare. See Core::X86::Msr::DR0_ADDR_MASK. AddrMask[11:0] qualifies the DR3 linear address instruction breakpoint, allowing the DR3 instruction breakpoint on a range of addresses in memory.

**MSRC001\_1027 [Address Mask For DR0 Breakpoints] (Core::X86::Msr::DR0\_ADDR\_MASK)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Support for DR0[31:12] is indicated by Core::X86::Cpuid::FeatureExtIdEcx[DataBreakpointExtension]. See Core::X86::Msr::DR1\_ADDR\_MASK.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1027

Bits	Description
63:32	Reserved.
31:0	<b>DR0: mask for DR0 linear address data breakpoint.</b> Read-write. Reset: 0000_0000h. 1=Exclude bit into address compare. 0=Include bit into address compare. See Core::X86::Msr::DR1_ADDR_MASK. This field qualifies the DR0 linear address data breakpoint, allowing the DR0 data breakpoint on a range of addresses in memory. AddrMask[11:0] qualifies the DR0 linear address instruction breakpoint, allowing the DR0 instruction breakpoint on a range of addresses in memory. DR0[31:12] is only valid for data breakpoints. The legacy DR0 breakpoint function is provided by DR0[31:0] == 0000_0000h). The mask bits are active high. DR0 is always used, and it can be used in conjunction with any debug function that uses DR0.

**MSRC001\_1030 [IBS Fetch Control] (Core::X86::Msr::IBS\_FETCH\_CTL)**

Reset: 0000\_0000\_0000\_0000h.

See 2.1.15 [Instruction Based Sampling (IBS)].

The IBS fetch sampling engine is described as follows:

- The periodic fetch counter is an internal 20-bit counter:
  - The periodic fetch counter [19:4] is set to IbsFetchCnt[19:4] and the periodic fetch counter [3:0] is set according to IbsRandEn when IbsFetchEn is changed from 0 to 1.
  - It increments for every fetch cycle that completes when IbsFetchEn == 1 and IbsFetchVal == 0.
    - The periodic fetch counter is undefined when IbsFetchEn == 0 or IbsFetchVal == 1.
  - When IbsFetchCnt[19:4] is read it returns the current value of the periodic fetch counter [19:4].
- When the periodic fetch counter reaches {IbsFetchMaxCnt[19:4],0h} and the selected instruction fetch completes or is aborted:
  - IbsFetchVal is set to 1.
    - Drivers can't assume that IbsFetchCnt[19:4] is 0 when IbsFetchVal == 1.
- The status of the operation is written to the IBS fetch registers (this register, Core::X86::Msr::IBS\_FETCH\_LINADDR and Core::X86::Msr::IBS\_FETCH\_PHYSADDR).
- An interrupt is generated as specified by Core::X86::Msr::IBS\_CTL. The interrupt service routine associated with this interrupt is responsible for saving the performance information stored in IBS execution registers.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1030

Bits	Description										
63:62	Reserved.										
61	<b>IbsFetchL3Miss: L3 cache miss for the sampled fetch.</b> Read-write, Volatile. Reset: 0. 1=The instruction fetch missed in the L3 Cache on the same CCX.										
60	<b>IbsFetchOcMiss: Op cache miss for the sampled fetch.</b> Read-write, Volatile. Reset: 0. 1=The Op Cache was not able to supply all bytes for the tagged fetch.										
59	<b>IbsL3MissOnly: Only L3 miss samples are reported.</b> Read-write, Volatile. Reset: 0. 1=An IBS interrupt is only created for fetch samples that had an L3 miss; Fetch samples without an L3 miss are discarded and the internal periodic fetch counter is reset to a pseudo random value between 0 and 15. Allows for filtering of Fetch IBS samples based on their L3Miss status.										
58	<b>IbsFetchL2Miss: L2 cache miss for the sampled fetch.</b> Read-write, Volatile. Reset: 0. 1=The instruction fetch missed in the L2 Cache. Qualified by (IbsFetchComp == 1).										
57	<b>IbsRandEn: random instruction fetch tagging enable.</b> Read-write. Reset: 0. 0=Bits[3:0] of the fetch counter are set to 0h when IbsFetchEn is set to start the fetch counter. 1=Bits[3:0] of the fetch counter are randomized when IbsFetchEn is set to start the fetch counter.										
56	<b>IbsL2TlbMiss: instruction cache L2TLB miss.</b> Read-write, Volatile. Reset: 0. 1=The instruction fetch missed in the L2 TLB.										
55	<b>IbsL1TlbMiss: instruction cache L1TLB miss.</b> Read-write, Volatile. Reset: 0. 1=The instruction fetch missed in the L1 TLB.										
54:53	<b>IbsL1TlbPgSz: instruction cache L1TLB page size.</b> Read-write, Volatile. Reset: 0h. Indicates the page size of the translation in the L1 TLB. This field is only valid if IbsPhyAddrValid == 1.										
<b>ValidValues:</b>											
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>4 KB</td></tr> <tr> <td>1h</td><td>2 MB</td></tr> <tr> <td>2h</td><td>1 GB</td></tr> <tr> <td>3h</td><td>16 KB (Coalesced 4 KB pages)</td></tr> </table>	Value	Description	0h	4 KB	1h	2 MB	2h	1 GB	3h	16 KB (Coalesced 4 KB pages)
Value	Description										
0h	4 KB										
1h	2 MB										
2h	1 GB										
3h	16 KB (Coalesced 4 KB pages)										
52	<b>IbsPhyAddrValid: instruction fetch physical address valid.</b> Read-write, Volatile. Reset: 0. 1=The physical address in Core::X86::Msr::IBS_FETCH_PHYSADDR and the IbsL1TlbPgSz field are valid for the instruction fetch.										
51	<b>IbsIcMiss: instruction cache miss.</b> Read-write, Volatile. Reset: 0. 1=The instruction fetch missed in either the instruction cache or the Op Cache.										



50	<b>IbsFetchComp: instruction fetch complete.</b> Read-write, Volatile. Reset: 0. 1=The instruction fetch completed and the data is available for use by the instruction decoder.
49	<b>IbsFetchVal: instruction fetch valid.</b> Read-write, Volatile. Reset: 0. 1=New instruction fetch data available. When this bit is set, the fetch counter stops counting and an interrupt is generated as specified by Core::X86::Msrr::IBS_CTL. This bit must be cleared for the fetch counter to start counting. When clearing this bit, software can write 0000h to IbsFetchCnt[19:4] to start the fetch counter at IbsFetchMaxCnt[19:4].
48	<b>IbsFetchEn: instruction fetch enable.</b> Read-write. Reset: 0. 1=Instruction fetch sampling is enabled.
47:32	<b>IbsFetchLat: instruction fetch latency.</b> Read-write, Volatile. Reset: 0000h. Indicates the number of clock cycles from when the instruction fetch was initiated to when the data was delivered to the core. If the instruction fetch is abandoned before the fetch completes, this field returns the number of clock cycles from when the instruction fetch was initiated to when the fetch was abandoned.
31:16	<b>IbsFetchCnt[19:4].</b> Read-write, Volatile. Reset: 0000h. Provides Read/Write access to bits[19:4] of the periodic fetch counter. Programming this field to a value greater than or equal to IbsFetchMaxCnt[19:4] results in undefined behavior.
15:0	<b>IbsFetchMaxCnt[19:4].</b> Read-write. Reset: 0000h. Specifies bits[19:4] of the maximum count value of the periodic fetch counter. Programming this field to 0000h and setting IbsFetchEn results in undefined behavior. Bits[3:0] of the maximum count are always 0000b.

#### MSRC001\_1031 [IBS Fetch Linear Address] (Core::X86::Msrr::IBS\_FETCH\_LINADDR)

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1031

Bits	Description
63:0	<b>IbsFetchLinAd: instruction fetch linear address.</b> Read-write, Volatile. Reset: 0000_0000_0000_0000h. Provides the linear address in canonical form for the tagged instruction fetch.

#### MSRC001\_1032 [IBS Fetch Physical Address] (Core::X86::Msrr::IBS\_FETCH\_PHYSADDR)

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1032

Bits	Description
63:48	Reserved.
47:0	<b>IbsFetchPhysAd: instruction fetch physical address.</b> Read-write, Volatile. Reset: 0000_0000_0000h. Provides the physical address for the tagged instruction fetch. The lower 12 bits are not modified by address translation, so they are always the same as the linear address. This field contains valid data only if Core::X86::Msrr::IBS_FETCH_CTL[IbsPhyAddrValid] is asserted. When nested paging is active, the reported physical address is the system physical address. This register reads zero for a guest with active IBS Virtualization.

**MSRC001\_1033 [IBS Execution Control] (Core::X86::Msr::IBS\_OP\_CTL)**

Reset: 0000\_0000\_0000\_0000h.

See 2.1.15 [Instruction Based Sampling (IBS)].

The IBS execution sampling engine is described as follows for IbsOpCntCtl == 1. If IbsOpCntCtl == 0 then references to "periodic op counter" mean "periodic cycle counter".

- The periodic op counter is an internal 27-bit counter:
  - It is set to IbsOpCurCnt[26:0] when IbsOpEn is changed from 0 to 1.
  - It increments every dispatched macro-op when IbsOpEn == 1 and IbsOpVal == 0.
    - The periodic op counter is undefined when IbsOpEn == 0 or IbsOpVal == 1.
  - When IbsOpCurCnt[26:0] is read then it returns the current value of the periodic op counter [26:0].
- When the periodic op counter reaches IbsOpMaxCnt:
  - The next dispatched op is tagged if IbsOpCntCtl == 1. A valid op in the next dispatched line is tagged if IbsOpCntCtl == 0. See IbsOpCntCtl.
  - The periodic op counter [26:7]=0; [6:0] is randomized by hardware.
- The periodic op counter is not modified when a tagged op is flushed.
- When a tagged op is retired and all sample data has been collected:
  - IbsOpVal is set to 1.
    - Drivers can't assume that IbsOpCurCnt is 0 when IbsOpVal == 1.
- The status of the operation is written to the IBS execution registers (this register, Core::X86::Msr::IBS\_OP\_RIP, Core::X86::Msr::IBS\_OP\_DATA, Core::X86::Msr::IBS\_OP\_DATA2, Core::X86::Msr::IBS\_OP\_DATA3, Core::X86::Msr::IBS\_DC\_LINADDR and Core::X86::Msr::IBS\_DC\_PHYSADDR).
- An interrupt is generated as specified by Core::X86::Msr::IBS\_CTL. The interrupt service routine associated with this interrupt is responsible for saving the performance information stored in IBS execution registers.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1033

Bits	Description
63	<b>IbsOpLatFltEn.</b> Read-write, Volatile. Reset: 0. Load latency filter enable.
62:59	<b>IbsOpLatThrsh.</b> Read-write, Volatile. Reset: 0h. If latency thresholding is enabled, the latency threshold is calculated as (IbsOpLatThrsh+1) * 128 and compared with the DCIBSDATA.DcMissLatency value.
58:32	<b>IbsOpCurCnt[26:0]: periodic op counter current count.</b> Read-write, Volatile. Reset: 000_0000h. Returns the current value of the periodic op counter.
31:27	Reserved.
26:20	<b>IbsOpMaxCnt[26:20]: periodic op counter maximum count.</b> Read-write. Reset: 00h. See IbsOpMaxCnt[19:4].
19	<b>IbsOpCntCtl: periodic op counter count control.</b> Read-write. Reset: 0. 0=Count clock cycles; a 1-of-4 round-robin counter selects an op in the next dispatch line; if the op pointed to by the round-robin counter is invalid, then the next younger valid op is selected. 1=Count dispatched ops; when a roll-over occurs, the counter is preloaded with a pseudorandom 7 bit value between 1 and 127.
18	<b>IbsOpVal: op sample valid.</b> Read-write, Volatile. Reset: 0. 1=New instruction execution data available; the periodic op counter is disabled from counting. An interrupt may be generated when this bit is set as specified by Core::X86::Msr::IBS_CTL[LvtOffset].
17	<b>IbsOpEn: op sampling enable.</b> Read-write. Reset: 0. 1=Instruction execution sampling enabled.
16	<b>IbsOpL3MissOnly.</b> Read-write. Reset: 0. 1=An IBS interrupt is only created for op samples that had an L3 miss. L3 miss in this context means the data came outside this core's CCX. A hit in another CCX's L3 is considered an L3 miss. Op samples that do not have an L3 miss are dropped and IbsOpCurCnt is reset to a pseudo random value between 0 and 127.
15:0	<b>IbsOpMaxCnt[19:4]: periodic op counter maximum count.</b> Read-write. Reset: 0000h. IbsOpMaxCnt[26:0] = {IbsOpMaxCnt[26:20], IbsOpMaxCnt[19:4], 0000b}. Specifies maximum count value of the periodic op counter. Bits[3:0] of the maximum count are always 0000b.



Valid Values:		
Value	Description	
0008h-0000h	Reserved.	
FFFFh-0009h	<Value> *16 Ops.	

#### MSRC001\_1034 [IBS Op RIP] (Core::X86::Msr::IBS\_OP\_RIP)

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1034

Bits	Description
63:0	<b>IbsOpRip.</b> Read-write, Volatile. Reset: 0000_0000_0000_0000h. 64 bit Segment offset (RIP) of the instruction that contains the tagged op.

#### MSRC001\_1035 [IBS Op Data] (Core::X86::Msr::IBS\_OP\_DATA)

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1035

Bits	Description
63:41	Reserved.
40	<b>IbsOpMicrocode.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation from microcode.
39	<b>IbsOpFuse: fused instruction op.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation was part of a fused instruction pair; IBS_OP_RIP reports the rIP of the older instruction within that pair. Support indicated by Core::X86::Cpuid::IbsIdEax[OpFuse].
38	<b>IbsRipInvalid: RIP is invalid.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation RIP is invalid. Support indicated by Core::X86::Cpuid::IbsIdEax[RipInvalidChk].
37	<b>IbsOpBrnRet: branch op retired.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation was a branch op that retired.
36	<b>IbsOpBrnMisp: mispredicted branch op.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation was a branch op that was mispredicted. Qualified by IbsOpBrnRet == 1.
35	<b>IbsOpBrnTaken: taken branch op.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation was a branch op that was taken. Qualified by IbsOpBrnRet == 1.
34	<b>IbsOpReturn: return op.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation was return op. Qualified by (IbsOpBrnRet == 1).
33:32	Reserved.
31:16	<b>IbsTagToRetCtr: op tag to retire count.</b> Read-write, Volatile. Reset: 0000h. This field returns the number of cycles from when the op was tagged to when the op was retired. This field is equal to IbsCompToRetCtr when the tagged op has zero-cycle latency.
15:0	<b>IbsCompToRetCtr: op completion to retire count.</b> Read-write, Volatile. Reset: 0000h. This field returns the number of cycles from when the op was completed to when the op was retired.

**MSRC001\_1036 [IBS Op Data 2] (Core::X86::Msr::IBS\_OP\_DATA2)**

Reset: 0000\_0000\_0000\_0000h.

Data is only valid for load operations that miss both the L1 data cache and the L2 cache.

If a load or store operation crosses a 64B boundary, the data returned in this register is for the lower of the two cache lines accessed.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1036

Bits	Description																										
63:8	Reserved.																										
7:6	<b>DataSrc[4:3]: Northbridge IBS request data source high bits.</b> Read-write, Volatile. Reset: 0h. High order bits for the DataSrc field. See DataSrcLo for a description of the valid values.																										
5	<b>NearFarCache_NearFar: Requests that return from any cache.</b> Read-write, Volatile. Reset: 0. 0=M State. 1=O State. Valid when the data source type is a cache.																										
4	<b>FarAny_NearFar: Requests that return from another NUMA node.</b> Read-write, Volatile. Reset: 0. 0=The request is serviced by the Northbridge in the same node as the requesting core. 1=The request is serviced by the Northbridge in a different NUMA node than the requesting core. Valid when DataSrc is non-zero.																										
3	Reserved.																										
2:0	<b>DataSrc[2:0]: Northbridge IBS request data source low bits.</b> Read-write. Reset: 0h. <b>Description:</b> Valid Values for {DataSrc[4:3], DataSrc[2:0]: }																										
	<table> <tr> <th>Values</th><th>Description</th></tr> <tr> <td>0h</td><td>No valid status.</td></tr> <tr> <td>1h</td><td>LocalCcx. Local L3 or different L2 in the same CCX.</td></tr> <tr> <td>2h</td><td>NearFarCache_Near. Requests that target the same NUMA node and return from another CCX's cache.</td></tr> <tr> <td>3h</td><td>DramIO_Near. Requests that target the same NUMA node and return from DRAM or MMIO.</td></tr> <tr> <td>4h</td><td>Reserved.</td></tr> <tr> <td>5h</td><td>NearFarCache_Far. Requests that target another NUMA node and return from another CCX's cache.</td></tr> <tr> <td>6h</td><td>LongLat_NearFar. Long-latency DIMM.</td></tr> <tr> <td>7h</td><td>DramIO_Far. Requests that target another NUMA node and return from DRAM or MMIO.</td></tr> <tr> <td>8h</td><td>Ext_NearFar. Extension Memory.</td></tr> <tr> <td>9h-Bh</td><td>Reserved.</td></tr> <tr> <td>Ch</td><td>Peer_NearFar. Coherent Memory of a different processor type.</td></tr> <tr> <td>Dh-1Fh</td><td>Reserved.</td></tr> </table>	Values	Description	0h	No valid status.	1h	LocalCcx. Local L3 or different L2 in the same CCX.	2h	NearFarCache_Near. Requests that target the same NUMA node and return from another CCX's cache.	3h	DramIO_Near. Requests that target the same NUMA node and return from DRAM or MMIO.	4h	Reserved.	5h	NearFarCache_Far. Requests that target another NUMA node and return from another CCX's cache.	6h	LongLat_NearFar. Long-latency DIMM.	7h	DramIO_Far. Requests that target another NUMA node and return from DRAM or MMIO.	8h	Ext_NearFar. Extension Memory.	9h-Bh	Reserved.	Ch	Peer_NearFar. Coherent Memory of a different processor type.	Dh-1Fh	Reserved.
Values	Description																										
0h	No valid status.																										
1h	LocalCcx. Local L3 or different L2 in the same CCX.																										
2h	NearFarCache_Near. Requests that target the same NUMA node and return from another CCX's cache.																										
3h	DramIO_Near. Requests that target the same NUMA node and return from DRAM or MMIO.																										
4h	Reserved.																										
5h	NearFarCache_Far. Requests that target another NUMA node and return from another CCX's cache.																										
6h	LongLat_NearFar. Long-latency DIMM.																										
7h	DramIO_Far. Requests that target another NUMA node and return from DRAM or MMIO.																										
8h	Ext_NearFar. Extension Memory.																										
9h-Bh	Reserved.																										
Ch	Peer_NearFar. Coherent Memory of a different processor type.																										
Dh-1Fh	Reserved.																										

**MSRC001\_1037 [IBS Op Data 3] (Core::X86::Msrr::IBS\_OP\_DATA3)**

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

Data in this register is only valid when either Core::X86::Msrr::IBS\_OP\_DATA3[IbsStOp] or Core::X86::Msrr::IBS\_OP\_DATA3[IbsLdOp] are set.

If a load or store operation crosses a 64B boundary, the data returned in this register is for the lower of the two cache lines accessed.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1037

Bits	Description																				
63:48	<b>IbsTlbRefillLat: L1 DTLB refill latency.</b> Read-write, Volatile. Reset: 0000h. The number of cycles from when a L1 DTLB refill is triggered by a tagged op to when the L1 DTLB fill has been completed. This field is only valid when Core::X86::Msrr::IBS_OP_DATA3[IbsDcPhyAddrValid] is set.																				
47:32	<b>IbsDcMissLat: data cache miss latency.</b> Read-write, Volatile. Reset: 0000h. Indicates the number of clock cycles from when a miss is detected in the data cache to when the data was delivered to the core. The value returned by this counter is not valid for non-load ops (Core::X86::Msrr::IBS_OP_DATA3[IbsLdOp]=0) or software prefetch ops (Core::X86::Msrr::IBS_OP_DATA3[IbsSwPf]=1).																				
31:26	<b>IbsOpDcMissOpenMemReqs: outstanding memory requests on DC fill.</b> Read-write, Volatile. Reset: 00h. The number of allocated, valid DC MABs when the MAB corresponding to a tagged DC miss op is deallocated. Includes the MAB allocated by the sampled op. 00000b=No information provided.																				
25:22	<b>IbsOpMemWidth: load/store size in bytes.</b> Read-write, Volatile. Reset: 0h. Report the number of bytes the load or store is attempting to access. <b>Valid Values:</b>																				
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>No information provided.</td></tr> <tr> <td>1h</td><td>Byte.</td></tr> <tr> <td>2h</td><td>Word.</td></tr> <tr> <td>3h</td><td>DW (4 Bytes).</td></tr> <tr> <td>4h</td><td>QW (8 Bytes).</td></tr> <tr> <td>5h</td><td>OW (16 Bytes).</td></tr> <tr> <td>6h</td><td>32 Bytes.</td></tr> <tr> <td>7h</td><td>64 Bytes.</td></tr> <tr> <td>Fh-8h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	No information provided.	1h	Byte.	2h	Word.	3h	DW (4 Bytes).	4h	QW (8 Bytes).	5h	OW (16 Bytes).	6h	32 Bytes.	7h	64 Bytes.	Fh-8h	Reserved.
Value	Description																				
0h	No information provided.																				
1h	Byte.																				
2h	Word.																				
3h	DW (4 Bytes).																				
4h	QW (8 Bytes).																				
5h	OW (16 Bytes).																				
6h	32 Bytes.																				
7h	64 Bytes.																				
Fh-8h	Reserved.																				
21	<b>IbsSwPf: software prefetch.</b> Read-write, Volatile. Reset: 0. 1=The op is a software prefetch.																				
20	<b>IbsL2Miss: L2 cache miss for the sampled operation.</b> Read-write, Volatile. Reset: 0. 1=The operation missed in the L2, regardless of whether the op initiated the request to the L2. This is not expected to be set for store operations to non-cacheable memory.																				
19	Reserved.																				
18	<b>IbsDcPhyAddrValid: data cache physical address valid.</b> Read-write, Volatile. Reset: 0. 1=The physical address in Core::X86::Msrr::IBS_DC_PHYSADDR is valid for the load or store operation.																				
17	<b>IbsDcLinAddrValid: data cache linear address valid.</b> Read-write, Volatile. Reset: 0. 1=The linear address in Core::X86::Msrr::IBS_DC_LINADDR is valid for the load or store operation.																				
16	<b>DcMissNoMabAlloc: DC miss with no MAB allocated.</b> Read-write, Volatile. Reset: 0. 1=The tagged load or store operation hit on an already allocated MAB.																				
15	<b>IbsDcLockedOp: locked operation.</b> Read-write, Volatile. Reset: 0. 1=Tagged load or store operation is a locked operation.																				
14	<b>IbsDcUcMemAcc: UC memory access.</b> Read-write, Volatile. Reset: 0. 1=Tagged load or store operation accessed uncacheable memory.																				
13	<b>IbsDcWcMemAcc: WC memory access.</b> Read-write, Volatile. Reset: 0. 1=Tagged load or store operation accessed write combining memory.																				
12:9	Reserved.																				

8	<b>IbsDcMisAcc: misaligned access.</b> Read-write, Volatile. Reset: 0. 1=The tagged load or store operation crosses a 64 byte address boundary.
7	<b>IbsDcMiss: data cache miss.</b> Read-write, Volatile. Reset: 0. 1=The cache line used by the tagged load or store was not present in the data cache. This is not expected to be set for store operations to non-cacheable memory.
6	Reserved.
5	<b>IbsDcL1TlbHit1G: data cache L1TLB hit in 1G page.</b> Read-write, Volatile. Reset: 0. 1=The physical address for the tagged load or store operation was present in a 1G page table entry in the data cache L1TLB.
4	<b>IbsDcL1TlbHit2M: data cache L1TLB hit in 2M page.</b> Read-write, Volatile. Reset: 0. 1=The physical address for the tagged load or store operation was present in a 2M page table entry in the data cache L1TLB.
3	<b>IbsDcL2TlbMiss: data cache L2TLB miss.</b> Read-write, Volatile. Reset: 0. 1=The physical address for the tagged load or store operation was not present in the data cache L2TLB.
2	<b>IbsDcL1tlbMiss: data cache L1TLB miss.</b> Read-write, Volatile. Reset: 0. 1=The physical address for the tagged load or store operation was not present in the data cache L1TLB.
1	<b>IbsStOp: store op.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation is a store operation.
0	<b>IbsLdOp: load op.</b> Read-write, Volatile. Reset: 0. 1=Tagged operation is a load operation.

#### MSRC001\_1038 [IBS DC Linear Address] (Core::X86::Msr::IBS\_DC\_LINADDR)

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1038

Bits	Description
63:0	<b>IbsDcLinAd.</b> Read-write, Volatile. Reset: 0000_0000_0000_0000h. Provides the linear address in canonical form for the tagged load or store operation. This field contains valid data only if Core::X86::Msr::IBS_OP_DATA3[IbsDcLinAddrValid] is asserted.

#### MSRC001\_1039 [IBS DC Physical Address] (Core::X86::Msr::IBS\_DC\_PHYSADDR)

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_1039

Bits	Description
63:48	Reserved.
47:0	<b>IbsDcPhysAd: load or store physical address.</b> Read-write, Volatile. Reset: 0000_0000_0000h. Provides the physical address for the tagged load or store operation. The lower 12 bits are not modified by address translation, so they are always the same as the linear address. For memory accesses, that cross a 64B boundary (Core::X86::Msr::IBS_OP_DATA3[IbsDcMisAcc] is set), this field always points to the first cache line for the access. Cache miss related information in Core::X86::Msr::IBS_OP_DATA3 or Core::X86::Msr::IBS_OP_DATA2 will be for one of the two cache lines accessed by the load or store operation. This field contains valid data only if Core::X86::Msr::IBS_OP_DATA3[IbsDcPhyAddrValid]=1 and Core::X86::Msr::IBS_OP_DATA3[IbsDcLinAddrValid]=1. When nested paging is active, the reported physical address is the system physical address. This register reads zero for a guest with active IBS Virtualization.

#### MSRC001\_103A [IBS Control] (Core::X86::Msr::IBS\_CTL)

Read, Error-on-write.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]; MSRC001\_103A

Bits	Description
63:9	Reserved.
8	<b>LvtOffsetVal: IBS Fetch and Op local vector table offset valid.</b> Read, Error-on-write. Reset: X.
7:4	Reserved.
3:0	<b>LvtOffset: IBS Fetch and Op local vector table offset.</b> Read, Error-on-write. Reset: Xh.

<b>MSRC001_103B [IBS Branch Target Address] (Core::X86::Msr::BP_IBSTGT_RIP)</b>	
Read-write, Volatile. Reset: 0000_0000_0000_0000h.	
Support for this register indicated by Core::X86::Cpuid::IbsIdEax[BrnTrgt].	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSRC001_103B	
Bits	Description
63:0	<b>IbsBrTarget.</b> Read-write, Volatile. Reset: 0000_0000_0000_0000h. The logical address in canonical form for the branch target. It only contains a valid value for branch instructions (Core::X86::Msr::IBS_OP_DATA[IbsOpBrnRet] == 1).
<b>MSRC001_103C [IBS Fetch Control Extended] (Core::X86::Msr::IC_IBS_EXTD_CTL)</b>	
Read-write, Volatile. Reset: 0000_0000_0000_0000h.	
Support for this register indicated by Core::X86::Cpuid::IbsIdEax[IbsFetchCtlExtd].	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]; MSRC001_103C	
Bits	Description
63:16	Reserved.
15:0	<b>IbsItlbRefillLat: ITLB Refill Latency for the sampled fetch, if there is a reload.</b> Read-write, Volatile. Reset: 0000h. The number of cycles when the fetch engine is stalled for an ITLB reload for the sampled fetch. If there is no reload, the latency is 0.

### 2.1.13.5 L3 MSRs - MSRC001\_1xxx

See 2.2.1 [L3 MSR Registers].

## 2.1.14 Performance Monitor Counters

### 2.1.14.1 RDPMC Assignments

There are six core performance event counters per thread, six performance events counters per L3 complex and sixteen Data Fabric performance events counters mapped to the RDPMC instruction as follows:

- The RDPMC[5:0] instruction accesses core events. See 2.1.14.5 [Core Performance Monitor Counters].
- The RDPMC[9:6, 1B:10] instruction accesses data fabric events.
- The RDPMC[F:A] instruction accesses L3 cache events. See 2.1.14.6 [L3 Cache Performance Monitor Counters].

### 2.1.14.2 Performance Measurement

This section contains AMD's recommended method for collecting microarchitecture performance common to software optimization. This may require combining multiple performance event selections. Table 18 [Guidance for Common Performance Statistics with Complex Event Selects] lists formulas for collecting common performance statistics.

- The term Event is the full value written to Core::X86::Msr::PERF\_CTL0..5.
  - Core PMC select bits [63:36, 31:16] are at the user's discretion, (i.e., they are not part of the event selection).
- The term L3Event is the full value written to Core::X86::Msr::ChL3PmcCfg.
- The term DFEvent is the full value written to Core::X86::Msr::DF\_PERF\_CTL.

Some UnitMask fields are not disclosed, but may be used by 2.1.14.2 [Performance Measurement].

*Table 18: Guidance for Common Performance Statistics with Complex Event Selects*

Description	Equation
-------------	----------

Branch Prediction	
Execution-Time Branch Misprediction Ratio (Non-Speculative).	Event[0x4300C3] / Event[0x4300C2]
Basic Caching	
All Data Cache Accesses	Event[0x430729]
All L2 Cache Accesses	Event[0x43F160] + Event[0x431F70] + Event[0x431F71] + Event[0x431F72]
L2 Cache Access from L1 Instruction Cache Miss (including prefetch)	Event[0x431060]
L2 Cache Access from L1 Data Cache Miss (including Prefetch)	Event[0x43E060]
L2 Cache Access from L2 Cache HWPF	Event[0x431F70] + Event[0x431F71] + Event[0x431F72]
All L2 Cache Misses	Event[0x430964] + Event[0x431F71] + Event[0x431F72]
L2 Cache Miss from L1 Instruction Cache Miss	Event[0x430164]
L2 Cache Miss from L1 Data Cache Miss	Event[0x430864]
L2 Cache Miss from L2 Cache HWPF	Event[0x431F71] + Event[0x431F72]
All L2 Cache Hits	Event[0x43f664] + Event[0x431f70]
L2 Cache Hit from L1 Instruction Cache Miss	Event[0x430664]
L2 Cache Hit from L1 Data Cache Miss	Event[0x43F064]
L2 Cache Hit from L2 Cache HWPF	Event[0x431F70]
L3 Cache Accesses	L3Event[0x0300C0000040FF04]
L3 Miss (includes cacheline state change requests)	L3Event[0x0300C00000400104]
Average L3 Cache Read Miss Latency (in nanoseconds)	L3Event[0x0303C00000403FAC]*10/ L3Event[0x0303C00000403FAD]
Op Cache (64B) Fetch Miss Ratio	Event[0x20043048F] / Event[0x20043078F]
Instruction Cache (32B) Fetch Miss Ratio	Event[0x10043188E] / Event[0x100431F8E]
Advanced Caching	
L1 Data Cache Fills from DRAM or IO in any NUMA node	Event[0x434844]
L1 Data Cache Fills from a different NUMA node	Event[0x435044]
L1 Data Cache Fills from within the same CCX	Event[0x430344]
L1 Data Cache Fills from another CCX cache in any NUMA node	Event[0x431444]
L1 Data Cache Fills All	Event[0x435F44]
Demand L1 Data Cache Fills from local L2	Event[0x430143]
Demand L1 Data Cache Fills from local L3 or different L2 in same CCX	Event[0x430243]
Demand L1 Data Cache Fills from another CCX cache in the same NUMA node	Event[0x430443]
Demand L1 Data Cache Fills from DRAM or MMIO in the same NUMA node	Event[0x430843]
Demand L1 Data Cache Fills from another CCX cache in a different NUMA node	Event[0x431043]
Demand L1 Data Cache Fills from Remote Memory or IO	Event[0x434043]
64B lines written per WCB close	Event[0x430150] / Event[0x432063]
TLBs	
L1 ITLB Misses	Event[0x430084] + Event[0x430785]

L2 ITLB Misses & Instruction page walk	Event[0x430785]
L1 DTLB Misses	Event[0x43FF45]
L2 DTLB Misses & Data page walk	Event[0x43F045]
All TLBs Flushed	Event[0x43FF78]
Stalls	
Macro-ops Dispatched	Event[0x4307AA]
Mixed SSE/AVX Stalls	Event[0x430E0E]
Macro-ops Retired	Event[0x4300C1]

### 2.1.14.3 Pipeline Utilization Analysis

Table 19: Guidance for Pipeline Utilization Analysis Statistics

Name	Description	Equation
Level 1		
Total Dispatch Slots	Up to 8 instructions can be dispatched in one cycle.	$8 * \text{Event}[430076]$
Frontend Bound	Fraction of dispatch slots that remained unused because the frontend did not supply enough instructions/ops.	$\text{Event}[1004301A0] / \text{Total Dispatch Slots}$
Bad Speculation	Fraction of dispatched ops that did not retire.	$(\text{Event}[4307AA] - \text{Event}[4300C1]) / \text{Total Dispatch Slots}$
Backend Bound	Fraction of dispatch slots that remained unused because of backend stalls.	$\text{Event}[100431EA0] / \text{Total Dispatch Slots}$
SMT contention	Fraction of unused dispatch slots because the other thread was selected.	$\text{Event}[1004360A0] / \text{Total Dispatch Slots}$
Retiring	Fraction of dispatch slots used by ops that retired.	$\text{Event}[4300C1] / \text{Total Dispatch Slots}$
Level 2		
Frontend Bound - Latency	Fraction of dispatch slots that remained unused because of a latency bottleneck in the frontend, such as Instruction Cache or ITLB misses.	$8 * \text{Event}[1064301A0] / \text{Total Dispatch Slots}$
Frontend Bound - BW	Fraction of dispatch slots that remained unused because of a bandwidth bottleneck in the frontend, such as decode bandwidth or Op Cache fetch bandwidth.	$\text{Event}[1004301A0] - (8 * \text{Event}[1064301A0]) / \text{Total Dispatch Slots}$
Bad Speculation – Mispredicts	Fraction of dispatched ops that were flushed due to branch mispredicts.	$\text{Bad Speculation} * \text{Event}[4300C3] / (\text{Event}[4300C3] + \text{Event}[43019F])$
Bad Speculation - Pipeline Restarts	Fraction of dispatched ops that were flushed due to pipeline restarts (resyncs).	$\text{Bad Speculation} * \text{Event}[43019F] / (\text{Event}[4300C3] + \text{Event}[430796])$
Backend Bound - Memory	Fraction of dispatched slots that remained unused because of stalls due to the memory subsystem.	$\text{Backend Bound} * (\text{Event}[43A2D6] / \text{Event}[4302D6])$
Backend Bound – CPU	Fraction of dispatched slots that remained unused because of stalls not related to the memory subsystem.	$\text{Backend Bound} * (1 - (\text{Event}[43A2D6] / \text{Event}[4302D6]))$



		Event[4302D6]))
Retiring - Fastpath	Fraction of dispatch slots used by fastpath ops that retired.	$\text{Retiring} * (\text{Event}[4300C1] - \text{Event}[1004300C2]) / \text{Event}[4300C1]$
Retiring - Microcode	Fraction of dispatch slots used by microcode ops that retired.	$\text{Retiring} * \text{Event}[1004300C2] / \text{Event}[4300C1]$

#### 2.1.14.4 Large Increment per Cycle Events

Table 20: PMC\_Definitions

Term	Description
<b>MergeEvent</b>	A PMC event that is capable of counter increments greater than 15, thus requiring merging a pair of even/odd performance monitors.

The maximum increment for a regular performance event is 15 (i.e., a 4-bit event). However some event types can have a larger increments every cycle.

An option is provided for merging a pair of even/odd performance monitors to acquire an accurate count. First the odd numbered Core::X86::MsR::PERF\_CTL0..5 is programmed with the event Core::X86::Pmc::Core::Merge (PMCxFF) with the enable bit (En) turned on and with the remaining bits off. Then the corresponding even numbered Core::X86::MsR::PERF\_CTL0..5 is programmed with the desired PMC event. Both the odd and even numbered counter need to be enabled in Core::X86::MsR::PerfCntrGlobalCtl for the merged counter to count. The performance monitor combines the count value to an 8-bit increment event and extends the counter to a 64-bit counter.

Software wanting to preload a value to a merged counter pair writes the high-order 16-bit value to the low-order 16 bits of the odd counter and then writes the low-order 48-bit value to the even counter. Reading the even counter of the merged counter pair returns the full 64-bit value.

If an even performance monitor is programmed with the event Core::X86::Pmc::Core::Merge the Read results are undetermined. If an even performance monitor is programmed with a non-merge-able event (i.e., a 4-bit event) while the corresponding odd performance monitor is programmed as Merge, the Read results are undetermined. When discontinuing use of a merged counter pair, clear the Merge event from the odd performance monitor.

#### 2.1.14.5 Core Performance Monitor Counters

This section provides the core performance counter events that may be selected through Core::X86::MsR::PERF\_CTL0[EventSelect[11:8],EventSelect[7:0],UnitMask]. See Core::X86::MsR::PERF\_CTR. See Core::X86::MsR::PERF\_LEGACY\_CTL0..3 and Core::X86::MsR::PERF\_LEGACY\_CTR.



### 2.1.14.5.1 Floating-Point (FP) Events

#### PMCx002 [FP retired x87 uops] (Core::X86::Pmc::Core::Retired\_x87\_FP\_Ops)

Read-write.	
Number of retired x87 arithmetic operations. Can be used to calculate x87 FLOPs.	
PMCx002	
Bits	Description
7:3	Reserved.
2	<b>DivSqrROps.</b> Read-write. x87 Divide or square root uops.
1	<b>MulOps.</b> Read-write. x87 Multiply uops.
0	<b>AddSubOps.</b> Read-write. x87 Add/subtract uops.

#### PMCx003 [FP retired SSE and AVX FLOPs] (Core::X86::Pmc::Core::Retired\_SSE\_AVX\_FLOPs)

Read-write.																	
Number of SSE and AVX floating point arithmetic operations retired. Number of arithmetic operations retired is dependent on number of uops retired, data size (scalar/128/256/512), data type (BF16/FP16/FP32/FP64) and type of operation (add/sub/mul/mac/...). Use MergeEvent feature for accurate results.																	
PMCx003																	
Bits	Description																
7:5	<b>FlopTypeSel.</b> Read-write. Mask for specifying FLOP type. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>All types.</td></tr> <tr> <td>1h</td><td>B Float 16.</td></tr> <tr> <td>2h</td><td>Scalar single.</td></tr> <tr> <td>3h</td><td>Packed single.</td></tr> <tr> <td>4h</td><td>Scalar double.</td></tr> <tr> <td>5h</td><td>Packed double.</td></tr> <tr> <td>7h-6h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	All types.	1h	B Float 16.	2h	Scalar single.	3h	Packed single.	4h	Scalar double.	5h	Packed double.	7h-6h	Reserved.
Value	Description																
0h	All types.																
1h	B Float 16.																
2h	Scalar single.																
3h	Packed single.																
4h	Scalar double.																
5h	Packed double.																
7h-6h	Reserved.																
4	Reserved.																
3	<b>MacFLOPs.</b> Read-write. Each MAC operation count as 2 FLOPs. bfloat MAC operations are not included in this event.																
2	<b>DivFLOPs.</b> Read-write. Divide/square root FLOPs. Does not provide a useful count without use of the MergeEvent feature.																
1	<b>MultFLOPs.</b> Read-write. Multiply FLOPs. Does not provide a useful count without use of the MergeEvent feature.																
0	<b>AddSubFLOPs.</b> Read-write. Add/subtract FLOPs. Does not provide a useful count without use of the MergeEvent feature.																

**PMCx008 [FP uops retired by size] (Core::X86::Pmc::Core::Retired\_FP\_uOps)**

Read-write.

Report number of FP uops retired by size. Can be used to determine how vectorized code is and how much MMX / x87 content is in the code.

PMCx008

Bits	Description
7:6	Reserved.
5	<b>Pack512uOpsRetired.</b> Read-write. Packed 512-bit uops retired.
4	<b>Pack256uOpsRetired.</b> Read-write. Packed 256-bit uops retired.
3	<b>Pack128uOpsRetired.</b> Read-write. Packed 128-bit uops retired.
2	<b>ScalaruOpsRetired.</b> Read-write. Scalar uops retired.
1	<b>MMXuOpsRetired.</b> Read-write. MMX uops retired.
0	<b>x87uOpsRetired.</b> Read-write. x87 uops retired.

**PMCx00A [FP uops retired sorted by vector or scalar] (Core::X86::Pmc::Core::FP\_Ops\_Retired)**

Read-write.

Number of FP uops retired of selected type sorted by vector (AVX/SSE packed) or scalar (x87, AVX/SSE scalar). Can be used to profile FP codes.

PMCx00A

Bits	Description																																		
7:4	<b>VectorFpOpType.</b> Read-write. select a vector FP uop type to count or 0 for none. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>None selected.</td></tr> <tr><td>1h</td><td>Add.</td></tr> <tr><td>2h</td><td>Subtract.</td></tr> <tr><td>3h</td><td>Multiply.</td></tr> <tr><td>4h</td><td>Multiply accumulate.</td></tr> <tr><td>5h</td><td>Divide.</td></tr> <tr><td>6h</td><td>Square root.</td></tr> <tr><td>7h</td><td>Compare.</td></tr> <tr><td>8h</td><td>Convert.</td></tr> <tr><td>9h</td><td>Blend.</td></tr> <tr><td>Ah</td><td>Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.</td></tr> <tr><td>Bh</td><td>Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.</td></tr> <tr><td>Ch</td><td>BFloat.</td></tr> <tr><td>Dh</td><td>Logical.</td></tr> <tr><td>Eh</td><td>Other uops not included in previous groups.</td></tr> <tr><td>Fh</td><td>Select all fp type uops.</td></tr> </table>	Value	Description	0h	None selected.	1h	Add.	2h	Subtract.	3h	Multiply.	4h	Multiply accumulate.	5h	Divide.	6h	Square root.	7h	Compare.	8h	Convert.	9h	Blend.	Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.	Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.	Ch	BFloat.	Dh	Logical.	Eh	Other uops not included in previous groups.	Fh	Select all fp type uops.
Value	Description																																		
0h	None selected.																																		
1h	Add.																																		
2h	Subtract.																																		
3h	Multiply.																																		
4h	Multiply accumulate.																																		
5h	Divide.																																		
6h	Square root.																																		
7h	Compare.																																		
8h	Convert.																																		
9h	Blend.																																		
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.																																		
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.																																		
Ch	BFloat.																																		
Dh	Logical.																																		
Eh	Other uops not included in previous groups.																																		
Fh	Select all fp type uops.																																		
3:0	<b>ScalarFpOpType.</b> Read-write. select scalar FP uop type to count or 0 for none. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>None selected.</td></tr> <tr><td>1h</td><td>Add.</td></tr> <tr><td>2h</td><td>Subtract.</td></tr> <tr><td>3h</td><td>Multiply.</td></tr> <tr><td>4h</td><td>Multiply accumulate.</td></tr> <tr><td>5h</td><td>Divide.</td></tr> <tr><td>6h</td><td>Square root.</td></tr> <tr><td>7h</td><td>Compare.</td></tr> <tr><td>8h</td><td>Convert.</td></tr> <tr><td>9h</td><td>Blend.</td></tr> <tr><td>Ah</td><td>Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.</td></tr> <tr><td>Bh</td><td>Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.</td></tr> <tr><td>Ch</td><td>BFloat.</td></tr> <tr><td>Dh</td><td>Logical.</td></tr> <tr><td>Eh</td><td>Other uops not included in previous groups.</td></tr> <tr><td>Fh</td><td>Select all fp type uops.</td></tr> </table>	Value	Description	0h	None selected.	1h	Add.	2h	Subtract.	3h	Multiply.	4h	Multiply accumulate.	5h	Divide.	6h	Square root.	7h	Compare.	8h	Convert.	9h	Blend.	Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.	Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.	Ch	BFloat.	Dh	Logical.	Eh	Other uops not included in previous groups.	Fh	Select all fp type uops.
Value	Description																																		
0h	None selected.																																		
1h	Add.																																		
2h	Subtract.																																		
3h	Multiply.																																		
4h	Multiply accumulate.																																		
5h	Divide.																																		
6h	Square root.																																		
7h	Compare.																																		
8h	Convert.																																		
9h	Blend.																																		
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.																																		
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.																																		
Ch	BFloat.																																		
Dh	Logical.																																		
Eh	Other uops not included in previous groups.																																		
Fh	Select all fp type uops.																																		

**PMCx00B [FP executed integer type uops sorted by vector or scalar] (Core::X86::Pmc::Core::INT\_Ops\_Retired)**

Read-write.

Number of integer uops executed in the FP retired of selected type sorted by vector (SSE/AVX) or scalar (MMX). Can be used to profile vector INT / MMX codes.

PMCx00B

Bits	Description																																		
7:4	<b>SseAvxOpType.</b> Read-write. select SSE/AVX vector INT uop type to count or 0 for none. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>None selected.</td></tr> <tr><td>1h</td><td>Add.</td></tr> <tr><td>2h</td><td>Subtract.</td></tr> <tr><td>3h</td><td>Multiply.</td></tr> <tr><td>4h</td><td>Multiply accumulate.</td></tr> <tr><td>5h</td><td>AES.</td></tr> <tr><td>6h</td><td>SHA.</td></tr> <tr><td>7h</td><td>Compare.</td></tr> <tr><td>8h</td><td>Convert or pack.</td></tr> <tr><td>9h</td><td>Shift or rotate.</td></tr> <tr><td>Ah</td><td>Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.</td></tr> <tr><td>Bh</td><td>Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.</td></tr> <tr><td>Ch</td><td>VNNI.</td></tr> <tr><td>Dh</td><td>Logical.</td></tr> <tr><td>Eh</td><td>Other uops not included in previous groups.</td></tr> <tr><td>Fh</td><td>Select all int type uops.</td></tr> </table>	Value	Description	0h	None selected.	1h	Add.	2h	Subtract.	3h	Multiply.	4h	Multiply accumulate.	5h	AES.	6h	SHA.	7h	Compare.	8h	Convert or pack.	9h	Shift or rotate.	Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.	Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.	Ch	VNNI.	Dh	Logical.	Eh	Other uops not included in previous groups.	Fh	Select all int type uops.
Value	Description																																		
0h	None selected.																																		
1h	Add.																																		
2h	Subtract.																																		
3h	Multiply.																																		
4h	Multiply accumulate.																																		
5h	AES.																																		
6h	SHA.																																		
7h	Compare.																																		
8h	Convert or pack.																																		
9h	Shift or rotate.																																		
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.																																		
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.																																		
Ch	VNNI.																																		
Dh	Logical.																																		
Eh	Other uops not included in previous groups.																																		
Fh	Select all int type uops.																																		
3:0	<b>MmxOpType.</b> Read-write. select MMX INT scalar uop type to count or 0 for none. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>None selected.</td></tr> <tr><td>1h</td><td>Add.</td></tr> <tr><td>2h</td><td>Subtract.</td></tr> <tr><td>3h</td><td>Multiply.</td></tr> <tr><td>4h</td><td>Multiply accumulate.</td></tr> <tr><td>5h</td><td>AES.</td></tr> <tr><td>6h</td><td>SHA.</td></tr> <tr><td>7h</td><td>Compare.</td></tr> <tr><td>8h</td><td>Convert or pack.</td></tr> <tr><td>9h</td><td>Shift or rotate.</td></tr> <tr><td>Ah</td><td>Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.</td></tr> <tr><td>Bh</td><td>Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.</td></tr> <tr><td>Ch</td><td>VNNI.</td></tr> <tr><td>Dh</td><td>Logical.</td></tr> <tr><td>Eh</td><td>Other uops not included in previous groups.</td></tr> <tr><td>Fh</td><td>Select all int type uops.</td></tr> </table>	Value	Description	0h	None selected.	1h	Add.	2h	Subtract.	3h	Multiply.	4h	Multiply accumulate.	5h	AES.	6h	SHA.	7h	Compare.	8h	Convert or pack.	9h	Shift or rotate.	Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.	Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.	Ch	VNNI.	Dh	Logical.	Eh	Other uops not included in previous groups.	Fh	Select all int type uops.
Value	Description																																		
0h	None selected.																																		
1h	Add.																																		
2h	Subtract.																																		
3h	Multiply.																																		
4h	Multiply accumulate.																																		
5h	AES.																																		
6h	SHA.																																		
7h	Compare.																																		
8h	Convert or pack.																																		
9h	Shift or rotate.																																		
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.																																		
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.																																		
Ch	VNNI.																																		
Dh	Logical.																																		
Eh	Other uops not included in previous groups.																																		
Fh	Select all int type uops.																																		

**PMCx00C [FP uops retired sorted by packed 128 or packed 256]  
(Core::X86::Pmc::Core::Packed\_FP\_Ops\_Retired)**

Read-write.

Number of FP uops retired of selected type sorted by 128-bit packed dest (XMM) or 256-bit packed dest (YMM). Can be used to profile FP codes.

PMCx00C

Bits	Description																																		
7:4	<b>Fp256OpType.</b> Read-write. select a 256-bit packed FP uop type to count or 0 for none. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>None selected.</td></tr> <tr><td>1h</td><td>Add.</td></tr> <tr><td>2h</td><td>Subtract.</td></tr> <tr><td>3h</td><td>Multiply.</td></tr> <tr><td>4h</td><td>Multiply accumulate.</td></tr> <tr><td>5h</td><td>Divide.</td></tr> <tr><td>6h</td><td>Square root.</td></tr> <tr><td>7h</td><td>Compare.</td></tr> <tr><td>8h</td><td>Convert.</td></tr> <tr><td>9h</td><td>Blend.</td></tr> <tr><td>Ah</td><td>Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.</td></tr> <tr><td>Bh</td><td>Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.</td></tr> <tr><td>Ch</td><td>BFloat.</td></tr> <tr><td>Dh</td><td>Logical.</td></tr> <tr><td>Eh</td><td>Other uops not included in previous groups.</td></tr> <tr><td>Fh</td><td>Select all fp type uops.</td></tr> </table>	Value	Description	0h	None selected.	1h	Add.	2h	Subtract.	3h	Multiply.	4h	Multiply accumulate.	5h	Divide.	6h	Square root.	7h	Compare.	8h	Convert.	9h	Blend.	Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.	Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.	Ch	BFloat.	Dh	Logical.	Eh	Other uops not included in previous groups.	Fh	Select all fp type uops.
Value	Description																																		
0h	None selected.																																		
1h	Add.																																		
2h	Subtract.																																		
3h	Multiply.																																		
4h	Multiply accumulate.																																		
5h	Divide.																																		
6h	Square root.																																		
7h	Compare.																																		
8h	Convert.																																		
9h	Blend.																																		
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.																																		
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.																																		
Ch	BFloat.																																		
Dh	Logical.																																		
Eh	Other uops not included in previous groups.																																		
Fh	Select all fp type uops.																																		
3:0	<b>Fp128OpType.</b> Read-write. select 128-bit packed FP uop type to count or 0 for none. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>None selected.</td></tr> <tr><td>1h</td><td>Add.</td></tr> <tr><td>2h</td><td>Subtract.</td></tr> <tr><td>3h</td><td>Multiply.</td></tr> <tr><td>4h</td><td>Multiply accumulate.</td></tr> <tr><td>5h</td><td>Divide.</td></tr> <tr><td>6h</td><td>Square root.</td></tr> <tr><td>7h</td><td>Compare.</td></tr> <tr><td>8h</td><td>Convert.</td></tr> <tr><td>9h</td><td>Blend.</td></tr> <tr><td>Ah</td><td>Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.</td></tr> <tr><td>Bh</td><td>Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.</td></tr> <tr><td>Ch</td><td>BFloat.</td></tr> <tr><td>Dh</td><td>Logical.</td></tr> <tr><td>Eh</td><td>Other uops not included in previous groups.</td></tr> <tr><td>Fh</td><td>Select all fp type uops.</td></tr> </table>	Value	Description	0h	None selected.	1h	Add.	2h	Subtract.	3h	Multiply.	4h	Multiply accumulate.	5h	Divide.	6h	Square root.	7h	Compare.	8h	Convert.	9h	Blend.	Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.	Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.	Ch	BFloat.	Dh	Logical.	Eh	Other uops not included in previous groups.	Fh	Select all fp type uops.
Value	Description																																		
0h	None selected.																																		
1h	Add.																																		
2h	Subtract.																																		
3h	Multiply.																																		
4h	Multiply accumulate.																																		
5h	Divide.																																		
6h	Square root.																																		
7h	Compare.																																		
8h	Convert.																																		
9h	Blend.																																		
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.																																		
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily thought to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.																																		
Ch	BFloat.																																		
Dh	Logical.																																		
Eh	Other uops not included in previous groups.																																		
Fh	Select all fp type uops.																																		

**PMCx00D [FP executed packed integer uops sorted by packed 128 or packed 256]  
(Core::X86::Pmc::Core::Packed\_INT\_Ops\_Retired)**

Read-write.

Number of integer uops executed in FP retired of selected type sorted by 128-bit packed dest (XMM) or 256-bit packed dest (YMM). Can be used to profile FP codes.

PMCx00D

Bits	Description																																		
7:4	<b>Int256OpType.</b> Read-write. select a 256-bit packed INT uop type to count or 0 for none. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>None selected.</td></tr> <tr><td>1h</td><td>Add.</td></tr> <tr><td>2h</td><td>Subtract.</td></tr> <tr><td>3h</td><td>Multiply.</td></tr> <tr><td>4h</td><td>Multiply accumulate.</td></tr> <tr><td>5h</td><td>AES.</td></tr> <tr><td>6h</td><td>SHA.</td></tr> <tr><td>7h</td><td>Compare.</td></tr> <tr><td>8h</td><td>Convert or pack.</td></tr> <tr><td>9h</td><td>Shift or rotate.</td></tr> <tr><td>Ah</td><td>Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.</td></tr> <tr><td>Bh</td><td>Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.</td></tr> <tr><td>Ch</td><td>VNNI.</td></tr> <tr><td>Dh</td><td>Logical.</td></tr> <tr><td>Eh</td><td>Other uops not included in previous groups.</td></tr> <tr><td>Fh</td><td>Select all int type uops.</td></tr> </table>	Value	Description	0h	None selected.	1h	Add.	2h	Subtract.	3h	Multiply.	4h	Multiply accumulate.	5h	AES.	6h	SHA.	7h	Compare.	8h	Convert or pack.	9h	Shift or rotate.	Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.	Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.	Ch	VNNI.	Dh	Logical.	Eh	Other uops not included in previous groups.	Fh	Select all int type uops.
Value	Description																																		
0h	None selected.																																		
1h	Add.																																		
2h	Subtract.																																		
3h	Multiply.																																		
4h	Multiply accumulate.																																		
5h	AES.																																		
6h	SHA.																																		
7h	Compare.																																		
8h	Convert or pack.																																		
9h	Shift or rotate.																																		
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.																																		
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.																																		
Ch	VNNI.																																		
Dh	Logical.																																		
Eh	Other uops not included in previous groups.																																		
Fh	Select all int type uops.																																		
3:0	<b>Int128OpType.</b> Read-write. select 128-bit packed INT uop type to count or 0 for none. <b>ValidValues:</b> <table> <tr> <th>Value</th><th>Description</th></tr> <tr><td>0h</td><td>None selected.</td></tr> <tr><td>1h</td><td>Add.</td></tr> <tr><td>2h</td><td>Subtract.</td></tr> <tr><td>3h</td><td>Multiply.</td></tr> <tr><td>4h</td><td>Multiply accumulate.</td></tr> <tr><td>5h</td><td>AES.</td></tr> <tr><td>6h</td><td>SHA.</td></tr> <tr><td>7h</td><td>Compare.</td></tr> <tr><td>8h</td><td>Convert or pack.</td></tr> <tr><td>9h</td><td>Shift or rotate.</td></tr> <tr><td>Ah</td><td>Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.</td></tr> <tr><td>Bh</td><td>Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.</td></tr> <tr><td>Ch</td><td>VNNI.</td></tr> <tr><td>Dh</td><td>Logical.</td></tr> <tr><td>Eh</td><td>Other uops not included in previous groups.</td></tr> <tr><td>Fh</td><td>Select all int type uops.</td></tr> </table>	Value	Description	0h	None selected.	1h	Add.	2h	Subtract.	3h	Multiply.	4h	Multiply accumulate.	5h	AES.	6h	SHA.	7h	Compare.	8h	Convert or pack.	9h	Shift or rotate.	Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.	Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.	Ch	VNNI.	Dh	Logical.	Eh	Other uops not included in previous groups.	Fh	Select all int type uops.
Value	Description																																		
0h	None selected.																																		
1h	Add.																																		
2h	Subtract.																																		
3h	Multiply.																																		
4h	Multiply accumulate.																																		
5h	AES.																																		
6h	SHA.																																		
7h	Compare.																																		
8h	Convert or pack.																																		
9h	Shift or rotate.																																		
Ah	Move. MOV* instructions will count as INT type, not FP type. In other words, PMCx00A, PMCx00C will not count MOV ops.																																		
Bh	Shuffle. Shuf uop counts may count for instructions that are not necessarily though to include shuffles. i.e. horizontal add, dot-product, and some MOV instructions.																																		
Ch	VNNI.																																		
Dh	Logical.																																		
Eh	Other uops not included in previous groups.																																		
Fh	Select all int type uops.																																		

**PMCx00E [FP Dispatch Faults] (Core::X86::Pmc::Core::FP\_Dispatch\_Faults)**

Read-write.

Number of FP dispatch faults triggered by type. Dispatch fill/spill faults occur when FP either does not have the data needed to operate on in its local registers (fill), or FP needs to empty out upper register data for proper SSE merging behavior when executing AVX code (spill).

PMCx00E

Bits	Description
7:4	Reserved.
3	<b>YmmSpillFault.</b> Read-write. YMM spill fault
2	<b>YmmFillFault.</b> Read-write. YMM fill fault
1	<b>XmmFillFault.</b> Read-write. XMM Fill fault
0	<b>x87FillFault.</b> Read-write. x87 Fill fault

**2.1.14.5.2 Load/Store (LS) Events****PMCx024 [Bad Status 2] (Core::X86::Pmc::Core::Bad\_Status\_2\_STLI)**

Read-write.

Store To Load Interlock (STLI) are loads that were unable to complete because of a possible match with an older store, and the older store could not do Store To Load Forwarding (STLF) for some reason.

PMCx024

Bits	Description
7:2	Reserved.
1	<b>StliOther.</b> Read-write. Store-to-load conflicts: A load was unable to complete due to a non-forwardable conflict with an older store. Most commonly, a load's address range partially but not completely overlaps with an uncompleted older store. Software can avoid this problem by using same-size and same-alignment loads and stores when accessing the same data. Vector/SIMD code is particularly susceptible to this problem; software should construct wide vector stores by manipulating vector elements in registers using shuffle/blend/swap instructions prior to storing to memory, instead of using narrow element-by-element stores.
0	Reserved.

**PMCx025 [Retired Lock Instructions] (Core::X86::Pmc::Core::Retired\_Lock\_Instructions)**

Read-write.

Counts retired atomic read-modify-write instructions with a LOCK prefix.

PMCx025

Bits	Description
7:5	Reserved.
4:0	<b>LockInstructions.</b> Read-write. Specifies type of lock instructions counted
<b>ValidValues:</b>	
<b>Value</b>	<b>Description</b>
00h	Reserved.
01h	BusLock: Non-cacheable or cacheline-misaligned lock.
1Eh-02h	Reserved.
1Fh	AnyLock: Counts all lock instructions.

**PMCx026 [Retired CLFLUSH Instructions] (Core::X86::Pmc::Core::CLFLUSH)**

Read-write.

The number of retired CLFLUSH instructions. This is a non-speculative event.

PMCx026

**Bits Description**

7:0 Reserved.

**PMCx027 [Retired CPUID Instructions] (Core::X86::Pmc::Core::CUID)**

Read-write.

The number of CPUID instructions retired.

PMCx027

**Bits Description**

7:0 Reserved.

**PMCx029 [LS Dispatch] (Core::X86::Pmc::Core::LS\_Dispatch)**

Read-write.

Counts the number of operations dispatched to the LS unit. Unit Masks events are ADDED.

PMCx029

**Bits Description**

7:3 Reserved.

2 **LdOpSt.** Read-write. Dispatch of a single op that performs a load from and store to the same memory address.1 **PureSt.** Read-write. Dispatch of a single op that performs a memory store.0 **PureLd.** Read-write. Dispatch of a single op that performs a memory load.**PMCx02B [SMIs Received] (Core::X86::Pmc::Core::SMI\_or\_SMM\_cycles)**

Reset: 00h.

Counts the number of System Management Interrupts (SMIs) received.

PMCx02B

**Bits Description**

7:0 Reserved.

**PMCx02C [Interrupts Taken] (Core::X86::Pmc::Core::Interrupts\_Taken)**

Read-write.

Counts the number of interrupts taken.

PMCx02C

**Bits Description**

7:1 Reserved.

0 **NumInterrupts.** Read-write. Number of interrupts taken. This event is also counted when UnitMask[7:0]=0.**PMCx035 [Store to Load Forward] (Core::X86::Pmc::Core::Store\_to\_Load\_Forward)**

Read-write.

Number of STLF hits.

PMCx035

**Bits Description**

7:0 Reserved.



**PMCx037 [Store Globally Visible Cancels 2] (Core::X86::Pmc::Core::Store\_Globally\_Visible\_Cancels\_2)**

Read-write.

Counts reasons why a Store Coalescing Buffer (SCB) commit is canceled.

PMCx037

Bits	Description
7:1	Reserved.
0	<b>OlderStVisibleDepCancel.</b> Read-write. Older SCB we are waiting on to become globally visible was unable to become globally visible.

**PMCx041 [LS MAB Allocates by Type] (Core::X86::Pmc::Core::LS\_MAB\_Allocates\_by\_Type)**

Read-write.

Counts when an LS pipe allocates a Miss Address Buffer (MAB) entry to make a miss request.

PMCx041

Bits	Description														
7	Reserved.														
6:0	<b>LsMabAllocation.</b> Read-write. <b>ValidValues:</b>														
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>06h-00h</td><td>Reserved.</td></tr> <tr> <td>07h</td><td>Load Store Allocations</td></tr> <tr> <td>08h</td><td>Hardware Prefetcher Allocations</td></tr> <tr> <td>0Eh-09h</td><td>Reserved.</td></tr> <tr> <td>0Fh</td><td>All Allocations</td></tr> <tr> <td>7Fh-10h</td><td>Reserved.</td></tr> </table>	Value	Description	06h-00h	Reserved.	07h	Load Store Allocations	08h	Hardware Prefetcher Allocations	0Eh-09h	Reserved.	0Fh	All Allocations	7Fh-10h	Reserved.
Value	Description														
06h-00h	Reserved.														
07h	Load Store Allocations														
08h	Hardware Prefetcher Allocations														
0Eh-09h	Reserved.														
0Fh	All Allocations														
7Fh-10h	Reserved.														

**PMCx043 [Demand Data Cache Fills by Data Source] (Core::X86::Pmc::Core::Demand\_DC\_Fills\_by\_Data\_Source)**

Read-write.

Counts fills into the DC that were initiated by demand ops, per data source.

PMCx043

Bits	Description
7	<b>AlternateMemories_NearFar.</b> Read-write. Requests that return from Extension Memory.
6	<b>DramIO_Far.</b> Read-write. Requests that target another NUMA node and return from DRAM or MMIO.
5	Reserved.
4	<b>NearFarCache_Far.</b> Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	<b>DramIO_Near.</b> Read-write. Requests that target the same NUMA node and return from DRAM or MMIO.
2	<b>NearFarCache_Near.</b> Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	<b>LocalCcx.</b> Read-write. Data returned from L3 or different L2 in the same CCX.
0	<b>LocalL2.</b> Read-write. Data returned from local L2.

**PMCx044 [Any Data Cache Fills by Data Source] (Core::X86::Pmc::Core::Any\_DC\_Fills\_by\_Data\_Source)**

Read-write.

Counts all fills into the DC, per data source.

PMCx044

Bits	Description
7	<b>AlternateMemories_NearFar</b> . Read-write. Requests that return from Extension Memory.
6	<b>DramIO_Far</b> . Read-write. Requests that target another NUMA node and return from DRAM or MMIO.
5	Reserved.
4	<b>NearFarCache_Far</b> . Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	<b>DramIO_Near</b> . Read-write. Requests that target the same NUMA node and return from DRAM or MMIO.
2	<b>NearFarCache_Near</b> . Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	<b>LocalCcx</b> . Read-write. Data returned from L3 or different L2 in the same CCX.
0	<b>LocalL2</b> . Read-write. Data returned from local L2.

**PMCx045 [L1 DTLB Reloads] (Core::X86::Pmc::Core::L1\_DTLB\_Reloads)**

Read-write.

Counts L1DTLB reloads

PMCx045

Bits	Description
7	<b>TlbReload1GL2Miss</b> . Read-write. DTLB reload to a 1G page that missed in the L2DTLB.
6	<b>TlbReload2ML2Miss</b> . Read-write. DTLB reload to a 2M page that missed in the L2DTLB.
5	<b>TlbReloadCoalescedPageMiss</b> . Read-write. DTLB reload to a coalesced page that missed in the L2DTLB.
4	<b>TlbReload4KL2Miss</b> . Read-write. DTLB reload to a 4K page that missed in the L2DTLB.
3	<b>TlbReload1GL2Hit</b> . Read-write. DTLB reload to a 1G page that hit in the L2DTLB.
2	<b>TlbReload2ML2Hit</b> . Read-write. DTLB reload to a 2M page that hit in the L2DTLB.
1	<b>TlbReloadCoalescedPageHit</b> . Read-write. DTLB reload to a coalesced page that hit in the L2DTLB.
0	<b>TlbReload4KL2Hit</b> . Read-write. DTLB reload to a 4K page that hit in the L2DTLB.

**PMCx047 [Misaligned Load Flows] (Core::X86::Pmc::Core::Misaligned\_Load\_Flows)**

Read-write.

The number of misaligned load flows.

PMCx047

Bits	Description
7:2	Reserved.
1	<b>MA4K</b> . Read-write. The number of 4KB misaligned (i.e., page crossing) loads or LdOpSt.
0	<b>MA64</b> . Read-write. The number of 64B misaligned (i.e., cacheline crossing) loads or LdOpSt.

**PMCx04B [Prefetch Instructions Dispatched] (Core::X86::Pmc::Core::Software\_Prefetch\_Dispatched)**

Read-write.

Software Prefetch Instructions Dispatched (speculative)

PMCx04B

Bits	Description
7:3	Reserved.
2	<b>PREFETCHNTA</b> . Read-write. PrefetchNTA instruction. See docAPM3 PREFETCHlevel.
1	<b>PREFETCHW</b> . Read-write. PrefetchW instruction. See docAPM3 PREFETCHlevel.
0	<b>PREFETCH</b> . Read-write. PrefetchT0, T1, and T2 instructions. See docAPM3 PREFETCHlevel.

**PMCx050 [Write Combining Buffer Close] (Core::X86::Pmc::Core::WCB\_Close)**

Read-write.

Counts events that cause a Write Combining Buffer (WCB) entry to close.

PMCx050

Bits	Description
7:1	Reserved.
0	<b>FullLine64B.</b> Read-write. All 64 bytes of the WCB entry have been written.

**PMCx052 [Ineffective Software Prefetches] (Core::X86::Pmc::Core::Ineffective\_Software\_Prefetches)**

Read-write.

The number of software prefetches that did not fetch data outside of the processor core.

PMCx052

Bits	Description
7:2	Reserved.
1	<b>MabHit.</b> Read-write. Software PREFETCH instruction saw a match on an already-allocated miss request.
0	<b>DcHit.</b> Read-write. Software PREFETCH instruction saw a DC hit.

**PMCx059 [Software Prefetch Data Cache Fills by Data Source] (Core::X86::Pmc::Core::Software\_Prefetch\_Data\_Cache\_Fills)**

Read-write.

Counts fills into the DC that were initiated by software prefetch instructions, per data source.

PMCx059

Bits	Description
7	<b>AlternateMemories_NearFar.</b> Read-write. Requests that return from Extension Memory.
6	<b>DramIO_Far.</b> Read-write. Requests that target another NUMA node and return from DRAM or MMIO.
5	Reserved.
4	<b>NearFarCache_Far.</b> Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	<b>DramIO_Near.</b> Read-write. Requests that target the same NUMA node and return from DRAM or MMIO.
2	<b>NearFarCache_Near.</b> Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	<b>LocalCcx.</b> Read-write. Data returned from L3 or different L2 in the same CCX.
0	<b>LocalL2.</b> Read-write. Data returned from local L2.

**PMCx05A [Hardware Prefetch Data Cache Fills by Data Source] (Core::X86::Pmc::Core::Hardware\_Prefetch\_Data\_Cache\_Fills)**

Read-write.

Counts fills into the DC that were initiated by hardware prefetches, per data source.

PMCx05A

Bits	Description
7	<b>AlternateMemories_NearFar.</b> Read-write. Requests that return from Extension Memory.
6	<b>DramIO_Far.</b> Read-write. Requests that target another NUMA node and return from DRAM or MMIO.
5	Reserved.
4	<b>NearFarCache_Far.</b> Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	<b>DramIO_Near.</b> Read-write. Requests that target the same NUMA node and return from DRAM or MMIO.
2	<b>NearFarCache_Near.</b> Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	<b>LocalCcx.</b> Read-write. Data returned from L3 or different L2 in the same CCX.
0	<b>LocalL2.</b> Read-write. Data returned from local L2.

**PMCx05F [Allocated DC misses] (Core::X86::Pmc::Core::Allocated\_DC\_misses)**

Read-write.

Counts the number of in-flight DC misses each cycle.

PMCx05F

**Bits Description**

7:0 Reserved.

**PMCx076 [Cycles Not in Halt] (Core::X86::Pmc::Core::Cycles\_Not\_in\_Halt)**

Read-write.

Counts cycles when the thread is not in a HALTED state

PMCx076

**Bits Description**

7:0 Reserved.

**PMCx078 [All TLB Flushes] (Core::X86::Pmc::Core::TLB\_Flush\_Events)**

Read-write.

TLB flush events.

PMCx078

**Bits Description**

7:0 All. Read-write. All TLB Flushes

**ValidValues:****Value Description**

FEh-00h Reserved.

FFh Counts all TLB Flushes

**PMCx120 [P0 Freq Cycles not in Halt] (Core::X86::Pmc::Core::P0\_frequency\_Cycles\_Not\_in\_Halt)**

Read-write.

Counts cycles not in Halt, at the P0 P-state frequency, regardless of the current Pstate.

PMCx120

**Bits Description**

7:1 Reserved.

0 **P0\_frequency\_Cycles\_Not\_in\_Halt.** Read-write. Counts at the P0 frequency (same as Core::X86::Msr::MPERF) when not in Halt.**2.1.14.5.3 Instruction Cache (IC) and Branch Prediction (BP) Events**

Note: All instruction cache events are speculative events unless specified otherwise.

**PMCx082 [Instruction Cache Refills From L2] (Core::X86::Pmc::Core::Instruction\_Cache\_Refills\_from\_L2)**

Read-write.

The number of 64 byte instruction cache lines fulfilled from the L2 cache.

PMCx082

**Bits Description**

7:0 Reserved.

**PMCx083 [Instruction Cache Refills from System]  
(Core::X86::Pmc::Core::Instruction\_Cache\_Refills\_from\_System)**

Read-write.

The number of 64 byte instruction cache line fulfilled from system memory or another cache.

PMCx083

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx084 [L1 ITLB Miss, L2ITLB Hit] (Core::X86::Pmc::Core::L1\_ITLB\_Miss\_L2\_ITLB\_Hit)**

Read-write.

The number of instruction fetches that miss in the L1 ITLB but hit in the L2 ITLB.

PMCx084

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx085 [L1 ITLB Miss, L2 ITLB Miss] (Core::X86::Pmc::Core::ITLB\_Reload\_from\_Page\_Table\_walk)**

Read-write.

The number of instruction fetches that miss in both the L1 ITLB and L2 ITLB.

PMCx085

Bits	Description
------	-------------

7:4	Reserved.
-----	-----------

3	<b>Coalesced_4k.</b> Read-write. Walk for >4k Coalesced page (implemented as 16k)
---	---

2	<b>walk_1G.</b> Read-write. Walk for 1G page
---	--

1	<b>walk_2M.</b> Read-write. Walk for 2M page
---	--

0	<b>walk_4K.</b> Read-write. Walk to 4k page
---	---

**PMCx08B [BP Pipe Correction or Cancel] (Core::X86::Pmc::Core::BP\_Correct)**

Reset: 00h.

The Branch Predictor flushed its own pipeline due to internal conditions such as a second level prediction structure. Does not count the number of bubbles caused by these internal flushes.

PMCx08B

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx08E [Variable Target Predictions] (Core::X86::Pmc::Core::Variable\_Target\_Predictions)**

Read-write.

The number of times a branch used the indirect predictor to make a prediction.

PMCx08E

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx091 [Early Redirects]  
(Core::X86::Pmc::Core::Decoder\_Overrides\_Existing\_Branch\_Prediction\_Speculative)**

Reset: 00h.

Number of times that an Early Redirect is sent to Branch Predictor. This happens when either the decoder or dispatch logic is able to detect that the Branch Predictor needs to be redirected.

PMCx091

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx094 [ITLB Instruction Fetch Hits] (Core::X86::Pmc::Core::ITLB\_Hits)**

Read-write.

The number of instruction fetches that hit in the L1ITLB.

PMCx094

Bits	Description
7:3	Reserved.
2	<b>IF1G.</b> Read-write. L1 Instruction TLB Hit (1G page size)
1	<b>IF2M.</b> Read-write. L1 Instruction TLB Hit (2M page size)
0	<b>IF4K.</b> Read-write. L1 Instruction TLB Hit (4k or 16k coalesced page size)

**PMCx09F [BP Redirects] (Core::X86::Pmc::Core::BP\_redirects)**

Read-write.

Counts redirects of the branch predictor. To support legacy software, counts both EX mispredict and resyncs when unit\_mask[7:0] is set to 0.

PMCx09F

Bits	Description
7:2	Reserved.
1	<b>ExRedir.</b> Read-write. Mispredict redirect from EX (execution-time)
0	<b>Resync.</b> Read-write. Resync redirect (Retire-time) from RT

**PMCx188 [Fetch IBS events] (Core::X86::Pmc::Core::Fetch\_IBS\_events)**

Read-write.

Counts significant Fetch IBS State transitions.

PMCx188

Bits	Description
7:5	Reserved.
4	<b>SampleVal.</b> Read-write. Counts the number of valid Fetch Instruction Based Sampling (fetch IBS) samples that were collected. Each valid sample also created an IBS interrupt.
3	<b>SampleFiltered.</b> Read-write. Counts the number of Fetch IBS tagged fetches that were discarded due to IBS filtering. When a tagged fetch is discarded the Fetch IBS facility will automatically tag a new fetch.
2	<b>SampleDiscarded.</b> Read-write. Counts when the Fetch IBS facility discards an IBS tagged fetch for reasons other than IBS filtering. When a tagged fetch is discarded the Fetch IBS facility will automatically tag a new fetch.
1	<b>FetchTagged.</b> Read-write. Counts the number of fetches tagged for Fetch IBS. Not all tagged fetches create an IBS interrupt and valid fetch sample.
0	Reserved.

**PMCx18E [IC Tag Hit and Miss Events] (Core::X86::Pmc::Core::IC\_Tag\_Hit\_Miss\_events)**

Read-write.

Counts the number of microtag and full tag events as selected by unit mask.

PMCx18E

Bits	Description
7:5	Reserved.
4:0	<b>IcAccessTypes.</b> Read-write. Instruction Cache accesses.
<b>ValidValues:</b>	
Value	Description
06h-00h	Reserved.
07h	Instruction Cache Hit.
17h-08h	Reserved.
18h	Instruction Cache Miss.
1Eh-19h	Reserved.
1Fh	All Instruction Cache Accesses.

**PMCx28F [Op Cache Hit or Miss] (Core::X86::Pmc::Core::Op\_Cache\_hit\_miss)**

Read-write.

Counts Op Cache micro-tag hit/miss events.

PMCx28F

Bits	Description
7:3	Reserved.
2:0	<b>OpCacheAccesses.</b> Read-write. OpCacheAccesses
<b>ValidValues:</b>	
Value	Description
2h-0h	Reserved.
3h	Op Cache Hit.
4h	Op Cache Miss.
6h-5h	Reserved.
7h	All Op Cache accesses.

**2.1.14.5.4 DE Events****PMCx0A9 [Op Queue Empty] (Core::X86::Pmc::Core::Dispatch\_Empty)**

Reset: 00h.

Cycles where the Op Queue is empty.

PMCx0A9

Bits	Description
7:0	Reserved.

**PMCx0AA [Source of Op Dispatched From Decoder]  
(Core::X86::Pmc::Core::Source\_of\_Op\_Dispatched\_From\_Decoder)**

Read-write.

Counts the number of ops dispatched from the decoder classified by op source.

PMCx0AA

Bits	Description
7:2	Reserved.
1	<b>Op_Cache.</b> Read-write. Count of ops dispatched from OpCache
0	<b>x86_decoder.</b> Read-write. Count of ops dispatched from x86 decoder

**PMCx0AB [Types of Ops Dispatched From Decoder]  
(Core::X86::Pmc::Core::Types\_of\_Ops\_Dispatched\_From\_Decoder)**

Read-write.

Counts the number of ops dispatched from the decoder classified by op type. The UnitMask value encodes which types of ops are counted.

PMCx0AB

Bits	Description												
7:5	Reserved.												
4:0	<b>DispOpType.</b> Read-write. DispOpType. <b>ValidValues:</b>												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>03h-00h</td><td>Reserved.</td></tr> <tr> <td>04h</td><td>Any FP dispatch.</td></tr> <tr> <td>07h-05h</td><td>Reserved.</td></tr> <tr> <td>08h</td><td>Any Integer dispatch.</td></tr> <tr> <td>1Fh-09h</td><td>Reserved.</td></tr> </table>	Value	Description	03h-00h	Reserved.	04h	Any FP dispatch.	07h-05h	Reserved.	08h	Any Integer dispatch.	1Fh-09h	Reserved.
Value	Description												
03h-00h	Reserved.												
04h	Any FP dispatch.												
07h-05h	Reserved.												
08h	Any Integer dispatch.												
1Fh-09h	Reserved.												

**PMCx0AE [Dynamic Tokens Dispatch Stall Cycles 1]  
(Core::X86::Pmc::Core::Dispatch\_Stall\_Cycles\_Dynamic\_Tokens\_Part\_1)**

Read-write.

Cycles where a dispatch group is valid but does not get dispatched due to a Token Stall. UnitMask bits select the stall types included in the count.

PMCx0AE

Bits	Description
7	Reserved.
6	<b>FPSchRsrcStall.</b> Read-write. FP NSQ token stall
5	Reserved.
4	<b>TakenBrnchBufferRsrc.</b> Read-write. taken branch buffer resource stall.
3	Reserved.
2	<b>StoreQueueRsrcStall.</b> Read-write. STQ Tokens unavailable
1	<b>LoadQueueRsrcStall.</b> Read-write. Load Queue Token Stall.
0	<b>IntPhyRegFileRsrcStall.</b> Read-write. Integer Physical Register File resource stall.



**PMCx0AF [Dynamic Tokens Dispatch Stall Cycles 2] (Core::X86::Pmc::Core::Dispatch\_Stall\_Cycles\_Dynamic\_Tokens\_Part\_2)**

Read-write.

Cycles where a dispatch group is valid but does not get dispatched due to a token stall. UnitMask bits select the stall types included in the count.

PMCx0AF

Bits	Description
7:6	Reserved.
5	<b>RetQ.</b> Read-write. Retire queue tokens unavailable
4:3	Reserved.
2	<b>EX_Flush_recovery.</b> Read-write. Integer Execution flush recovery pending
1	<b>AGTokens.</b> Read-write. Agen tokens unavailable
0	<b>ALTokens.</b> Read-write. ALU tokens unavailable

**PMCx1A0 [No\_Dispatch\_per\_Slot] (Core::X86::Pmc::Core::No\_Dispatch\_per\_Slot)**

Read-write.

Counts the number of dispatch slots (each cycle) that remained unused for reasons selected by UnitMask.

PMCx1A0

Bits	Description																
7:0	<b>StallReason.</b> Read-write. <b>ValidValues:</b>																
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>01h</td><td>Counts dispatch slots left empty because the front-end did not supply ops.</td></tr> <tr> <td>1Dh-02h</td><td>Reserved.</td></tr> <tr> <td>1Eh</td><td>Counts ops unable to dispatch due to back-end stalls.</td></tr> <tr> <td>5Fh-1Fh</td><td>Reserved.</td></tr> <tr> <td>60h</td><td>Counts ops unable to dispatch because the dispatch cycle was granted to the other SMT thread.</td></tr> <tr> <td>FFh-61h</td><td>Reserved.</td></tr> </table>	Value	Description	00h	Reserved.	01h	Counts dispatch slots left empty because the front-end did not supply ops.	1Dh-02h	Reserved.	1Eh	Counts ops unable to dispatch due to back-end stalls.	5Fh-1Fh	Reserved.	60h	Counts ops unable to dispatch because the dispatch cycle was granted to the other SMT thread.	FFh-61h	Reserved.
Value	Description																
00h	Reserved.																
01h	Counts dispatch slots left empty because the front-end did not supply ops.																
1Dh-02h	Reserved.																
1Eh	Counts ops unable to dispatch due to back-end stalls.																
5Fh-1Fh	Reserved.																
60h	Counts ops unable to dispatch because the dispatch cycle was granted to the other SMT thread.																
FFh-61h	Reserved.																

**PMCx1A2 [Dispatch Additional Resource Stalls] (Core::X86::Pmc::Core::Additional\_Resource\_Stalls)**

Read-write.

This PMC event counts additional resource stalls that are not captured by Dispatch\_Stall\_Cycle\_Dynamic\_Tokens\_Part\_1 or Dispatch\_Stall\_Cycles\_Dynamic\_Tokens\_Part\_2.

PMCx1A2

Bits	Description								
7:0	<b>Stall.</b> Read-write. <b>ValidValues:</b>								
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>2Fh-00h</td><td>Reserved.</td></tr> <tr> <td>30h</td><td>Counts additional cycles dispatch is stalled due to the lack of dispatch resources.</td></tr> <tr> <td>FFh-31h</td><td>Reserved.</td></tr> </table>	Value	Description	2Fh-00h	Reserved.	30h	Counts additional cycles dispatch is stalled due to the lack of dispatch resources.	FFh-31h	Reserved.
Value	Description								
2Fh-00h	Reserved.								
30h	Counts additional cycles dispatch is stalled due to the lack of dispatch resources.								
FFh-31h	Reserved.								

**PMCxFFF [Merge] (Core::X86::Pmc::Core::Merge)**

See 2.1.14.4 [Large Increment per Cycle Events].

PMCxFFF

**Bits Description**

7:0 Reserved.

**2.1.14.5.5 EX (SC) Events****PMCx0C0 [Retired Instructions] (Core::X86::Pmc::Core::Retired\_Instructions)**

Read-write.

The number of instructions retired.

PMCx0C0

**Bits Description**

7:0 Reserved.

**PMCx0C1 [Retired Macro-Ops] (Core::X86::Pmc::Core::Retired\_Macro\_Ops)**

Read-write.

The number of macro-ops retired.

PMCx0C1

**Bits Description**

7:0 Reserved.

**PMCx0C2 [Retired Branch Instructions] (Core::X86::Pmc::Core::Retired\_Branch\_Instructions)**

Read-write.

The number of branch instructions retired. This includes all types of architectural control flow changes, including exceptions and interrupts.

PMCx0C2

**Bits Description**

7:0 Reserved.

**PMCx0C3 [Retired Branch Instructions Mispredicted.]  
(Core::X86::Pmc::Core::Retired\_Branch\_Instructions\_Mispredicted)**

Read-write.

The number of retired branch instructions, that were mispredicted. Note that only EX mispredicts are counted.

PMCx0C3

**Bits Description**

7:0 Reserved.

**PMCx0C4 [Retired Taken Branch Instructions] (Core::X86::Pmc::Core::Retired\_Taken\_Branch\_Instructions)**

Read-write.

The number of taken branches that were retired. This includes all types of architectural control flow changes, including exceptions and interrupts.

PMCx0C4

**Bits Description**

7:0 Reserved.

**PMCx0C5 [Retired Taken Branch Instructions Mispredicted.]**  
**(Core::X86::Pmc::Core::Retired\_Taken\_Branch\_Instructions\_Mispredicted)**

Read-write.

The number of retired taken branch instructions that were mispredicted. Note that only EX mispredicts are counted.

PMCx0C5

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx0C6 [Retired Far Control Transfers] (Core::X86::Pmc::Core::Retired\_Far\_Control\_Transfers)**

Read-write.

The number of far control transfers retired including far call/jump/return, IRET, SYSCALL and SYSRET, plus exceptions and interrupts. Far control transfers are not subject to branch prediction.

PMCx0C6

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx0C8 [Retired Near Return Branch Instructions]**  
**(Core::X86::Pmc::Core::Retired\_Near\_Return\_Branch\_Instructions)**

Read-write.

The number of near return instructions (RET [C3] or RET Iw [C2]) retired.

PMCx0C8

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx0C9 [Retired Near Return Branch Instructions Mispredicted]**  
**(Core::X86::Pmc::Core::Retired\_Near\_Return\_Branch\_Instructions\_Mispredicted)**

Read-write.

The number of near returns retired that were not correctly predicted by the return address predictor. Each such mispredict incurs the same penalty as a mispredicted conditional branch instruction. Note that only EX mispredicts are counted .

PMCx0C9

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx0CA [Retired Indirect Branch Instructions Mispredicted]**  
**(Core::X86::Pmc::Core::Retired\_Indirect\_Branch\_Instructions\_Mispredicted)**

Read-write.

The number of indirect branches retired that were not correctly predicted. Each such mispredict incurs the same penalty as a mispredicted conditional branch instruction. Note that only EX mispredicts are counted .

PMCx0CA

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx0CB [Retired MMX FP Instructions] (Core::X86::Pmc::Core::Retired\_MMX\_FP\_Instructions)**

Read-write.

The number of MMX, SSE or x87 instructions retired. The UnitMask allows the selection of the individual classes of instructions as given in the table. Each increment represents one complete instruction. Since this event includes non-numeric instructions it is not suitable for measuring MFLOPs

PMCx0CB

Bits	Description
7:3	Reserved.
2	SSE. Read-write. SSE instructions (SSE, SSE2, SSE3, SSSE3, SSE4A, SSE41, SSE42, AVX).
1	MMX. Read-write. MMX instructions
0	X87. Read-write. x87 instructions

**PMCx0CC [Retired Indirect Branch Instructions]  
(Core::X86::Pmc::Core::Retired\_Indirect\_Branch\_Instructions)**

Read-write.

The number of indirect branches retired.

PMCx0CC

Bits	Description
7:0	Reserved.

**PMCx0D1 [Retired Conditional Branch Instructions]  
(Core::X86::Pmc::Core::Retired\_Conditional\_Branch\_Instructions)**

Read-write.

Count of conditional branch instructions that retired

PMCx0D1

Bits	Description
7:0	Reserved.

**PMCx0D3 [Div Cycles Busy count] (Core::X86::Pmc::Core::Div\_Cycles\_Busy\_count)**

Read-write.

Counts cycles when the divider is busy

PMCx0D3

Bits	Description
7:0	Reserved.

**PMCx0D4 [Div Op Count] (Core::X86::Pmc::Core::Div\_Op\_Count)**

Read-write.

Counts number of divide ops

PMCx0D4

Bits	Description
7:0	Reserved.

**PMCx0D6 [Cycles with no retire] (Core::X86::Pmc::Core::Cycles\_With\_No\_Retire)**

Read-write.

This event counts cycles when the hardware thread does not retire any ops for reasons selected by UnitMask[4:0]. UnitMask events [4:0] are mutually exclusive. If multiple reasons apply for a given cycle, the lowest numbered UnitMask event is counted.

PMCx0D6

Bits	Description										
7:5	<b>CompletionFilter.</b> Read-write. <b>ValidValues:</b>										
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>0h</td><td>Load and ALU completion is considered for UnitMask[1]: NotComplete events.</td></tr> <tr> <td>4h-1h</td><td>Reserved.</td></tr> <tr> <td>5h</td><td>Only missing Load completion is considered for UnitMask[1]: NotComplete events.</td></tr> <tr> <td>7h-6h</td><td>Reserved.</td></tr> </table>	Value	Description	0h	Load and ALU completion is considered for UnitMask[1]: NotComplete events.	4h-1h	Reserved.	5h	Only missing Load completion is considered for UnitMask[1]: NotComplete events.	7h-6h	Reserved.
Value	Description										
0h	Load and ALU completion is considered for UnitMask[1]: NotComplete events.										
4h-1h	Reserved.										
5h	Only missing Load completion is considered for UnitMask[1]: NotComplete events.										
7h-6h	Reserved.										
4	<b>ThreadNotSelected.</b> Read-write. The number cycles where ops could have retired (i.e. did not fall into the sub-events [0]...[3]) but did not retire because the thread arbitration did not select the thread for retire.										
3	<b>Other.</b> Read-write. The number of cycles where ops could have retired (self and older ops are complete), but were stopped from retirement for other reasons: retire breaks, traps, faults, etc.										
2	Reserved.										
1	<b>NotCompleteSelf.</b> Read-write. The number of cycles where the oldest retire slot did not have its completion bits set.										
0	<b>Empty.</b> Read-write. The number of cycles when there were no valid ops in the retire queue. This may be caused by front-end bottlenecks or pipeline redirects.										

**PMCx1C1 [Retired Microcoded Instructions] (Core::X86::Pmc::Core::Retired\_Microcoded\_Instructions)**

Read-write.

The number of retired microcoded instructions.

PMCx1C1

Bits	Description
7:0	Reserved.

**PMCx1C2 [Retired Microcode Ops] (Core::X86::Pmc::Core::Retired\_Microcode\_Ops)**

Read-write.

The number of microcode ops that have retired.

PMCx1C2

Bits	Description
7:0	Reserved.

**PMCx1C7 [Retired Conditional Branch Instructions Mispredicted] (Core::X86::Pmc::Core::Retired\_Conditional\_Branch\_Instructions\_Mispredicted)**

Read-write.

The number of retired conditional branch instructions that were not correctly predicted because of a branch direction mismatch.

PMCx1C7

Bits	Description
7:0	Reserved.

**PMCx1C8 [Retired Unconditional Branch Instructions Mispredicted]  
(Core::X86::Pmc::Core::Retired\_Unconditional\_Branch\_Instructions\_Mispredicted)**

Read-write.

The number of retired unconditional indirect branch instructions that were mispredicted.

PMCx1C8

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx1C9 [Retired Unconditional Branch Instructions]  
(Core::X86::Pmc::Core::Retired\_Unconditional\_Branch\_Instructions)**

Read-write.

PMCx1C9

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

**PMCx1CF [Tagged IBS Ops] (Core::X86::Pmc::Core::Tagged\_IBS\_Ops)**

Read-write.

Counts Op IBS related events

PMCx1CF

Bits	Description
------	-------------

7:3	Reserved.
-----	-----------

2	<b>IbsCountRollover.</b> Read-write. Number of times an op could not be tagged by IBS because of a previous tagged op that has not yet signaled interrupt.
---	--

1	<b>IbsTaggedOpsRet.</b> Read-write. Number of Ops tagged by IBS that retired
---	--

0	<b>IbsTaggedOps.</b> Read-write. Number of Ops tagged by IBS
---	--

**PMCx1D0 [Retired Fused Instructions] (Core::X86::Pmc::Core::Retired\_fused\_instructions)**

Reset: 00h.

Counts retired fused instructions.

PMCx1D0

Bits	Description
------	-------------

7:0	Reserved.
-----	-----------

## 2.1.14.5.6 L2 Cache Events

**PMCx060 [Requests to L2 Group1] (Core::X86::Pmc::L2::L2RequestG1)**

Read-write.

All L2 Cache Requests (Breakdown 1 - Common)

PMCx060

Bits	Description
------	-------------

7	<b>RdBlkL.</b> Read-write. Data Cache Reads (including hardware and software prefetch).
---	---

6	<b>RdBlkX.</b> Read-write. Data Cache Stores
---	--

5	<b>LsRdBlkC_S.</b> Read-write. Data Cache Shared Reads
---	--

4	<b>CacheableIcRead.</b> Read-write. Instruction Cache Reads.
---	--

3	Reserved.
---	-----------

2	<b>LsPrefetchL2Cmd.</b> Read-write.
---	-------------------------------------

1	<b>L2HwPf: L2 Prefetcher.</b> Read-write. All prefetches accepted by L2 pipeline, hit or miss. Types of PF and L2 hit/miss broken out in a separate perfmon event
---	---

0	<b>Group2.</b> Read-write. MiscRequests. Read-write. Various Noncacheable requests. Non-cached Data Reads, Non-cached Instruction Reads, Self-modifying code checks.
---	--

**PMCx061 [Requests to L2 Group2] (Core::X86::Pmc::L2::L2RequestG2)**

Read-write.

All L2 Cache Requests (Breakdown 2 - Rare).

PMCx061

Bits	Description
7	Reserved.
6	<b>LsRdSized</b> . Read-write. LS sized read, coherent non-cacheable.
5	<b>LsRdSizedNC</b> . Read-write. LS sized read, non-coherent, non-cacheable.
4:0	Reserved.

**PMCx063 [Write Combining Buffer Requests] (Core::X86::Pmc::L2::L2WcbReq)**

Read-write.

Write Combining Buffer operations. For information on Write Combining see docAPM2 sections: Memory System, Memory Types, Buffering and Combining Memory Writes.

PMCx063

Bits	Description
7:6	Reserved.
5	<b>WcbClose</b> . Read-write. Write Combining Buffer close
4:0	Reserved.

**PMCx064 [Core to L2 Cacheable Request Access Status] (Core::X86::Pmc::L2::L2CacheReqStat)**

Read-write.

L2 Cache Request Outcomes (not including L2 Prefetch).

PMCx064

Bits	Description
7	<b>LsRdBlkCS: Data Cache Shared Read Hit in L2</b> . Read-write. LsRdBlkCS
6	<b>LsRdBlkLHitX: Data Cache Read Hit in L2</b> . Read-write. Modifiable
5	<b>LsRdBlkLHitS: Data Cache Read Hit Non-Modifiable Line in L2</b> . Read-write.
4	<b>LsRdBlkX: Data Cache Store Hit in L2</b> . Read-write.
3	<b>LsRdBlkC: Data Cache Req Miss in L2</b> . Read-write.
2	<b>IcFillHitX: Instruction Cache Hit Modifiable Line in L2</b> . Read-write. IcFillHitX
1	<b>IcFillHitS: Instruction Cache Hit Non-Modifiable Line in L2</b> . Read-write.
0	<b>IcFillMiss: Instruction Cache Req Miss in L2</b> . Read-write. IcFillMiss

**PMCx070 [L2 Prefetch Hit in L2] (Core::X86::Pmc::L2::L2PfHitL2)**

Read-write.

Counts all L2 prefetches accepted by L2 pipeline which hit in the L2 cache.

PMCx070

Bits	Description														
7:0	<b>Prefetches</b> . Read-write.														
<b>ValidValues:</b>															
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>1Eh-00h</td><td>Reserved.</td></tr> <tr> <td>1Fh</td><td>Counts requests generated from L2 Hardware Prefetchers.</td></tr> <tr> <td>DFh-20h</td><td>Reserved.</td></tr> <tr> <td>E0h</td><td>Counts requests generated from L1 DC Hardware Prefetchers.</td></tr> <tr> <td>FEh-E1h</td><td>Reserved.</td></tr> <tr> <td>FFh</td><td>Counts requests generated from L1 DC and L2 Hardware Prefetchers.</td></tr> </table>	Value	Description	1Eh-00h	Reserved.	1Fh	Counts requests generated from L2 Hardware Prefetchers.	DFh-20h	Reserved.	E0h	Counts requests generated from L1 DC Hardware Prefetchers.	FEh-E1h	Reserved.	FFh	Counts requests generated from L1 DC and L2 Hardware Prefetchers.
Value	Description														
1Eh-00h	Reserved.														
1Fh	Counts requests generated from L2 Hardware Prefetchers.														
DFh-20h	Reserved.														
E0h	Counts requests generated from L1 DC Hardware Prefetchers.														
FEh-E1h	Reserved.														
FFh	Counts requests generated from L1 DC and L2 Hardware Prefetchers.														

**PMCx071 [L2 Prefetcher Hits in L3] (Core::X86::Pmc::L2::L2PfMissL2HitL3)**

Read-write.

Counts all L2 prefetches accepted by the L2 pipeline which miss the L2 cache and hit the L3.

PMCx071

Bits	Description
7:0	<b>Prefetches.</b> Read-write. L2Stream
<b>ValidValues:</b>	
Value	Description
1Eh-00h	Reserved.
1Fh	Counts requests generated from L2 Hardware Prefetchers.
DFh-20h	Reserved.
E0h	Counts requests generated from L1 DC Hardware Prefetchers.
FEh-E1h	Reserved.
FFh	Counts requests generated from L1 DC and L2 Hardware Prefetchers.

**PMCx072 [L2 Prefetcher Misses in L3] (Core::X86::Pmc::L2::L2PfMissL2L3)**

Read-write.

Counts all L2 prefetches accepted by the L2 pipeline which miss the L2 and the L3 caches

PMCx072

Bits	Description
7:0	<b>Prefetches.</b> Read-write. L2Stream
<b>ValidValues:</b>	
Value	Description
1Eh-00h	Reserved.
1Fh	Counts requests generated from L2 Hardware Prefetchers.
DFh-20h	Reserved.
E0h	Counts requests generated from L1 DC Hardware Prefetchers.
FEh-E1h	Reserved.
FFh	Counts requests generated from L1 DC and L2 Hardware Prefetchers.



**PMCx165 [L2 Fill Response Source] (Core::X86::Pmc::L2::L2FillRspSrc)**

Read-write.

Counts fill responses based on their source. Selecting an event mask of 0xfe will count all L3 responses.

This will count all L3 responses to fill requests.

This event is similar to LS PMC 0x44

PMCx165

Bits	Description
7	<b>AlternateMemories_NearFar</b> . Read-write. "Requests that return from Extension Memory"
6	<b>DramIO_Far</b> . Read-write. Requests that target another NUMA node and return from either DRAM or MMIO from another NUMA node, either from the same or different NUMA node.
5	Reserved.
4	<b>NearFarCache_Far</b> . Read-write. Requests that target another NUMA node and return from another CCX's cache.
3	<b>DramIO_Near</b> . Read-write. Requests that target the same NUMA node and return from either DRAM or MMIO from the same NUMA node.
2	<b>NearFarCache_Near</b> . Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	<b>LocalCcx</b> . Read-write. Data returned from L3 or different L2 in the same CCX.
0	Reserved.

**2.1.14.6 L3 Cache Performance Monitor Counters**

This section provides the core performance counter events that may be selected through Core::X86::Msr::ChL3PmcCfg.

- When in non-SMT mode, thread 0 must be selected for events that don't ignore ThreadMask.

**2.1.14.6.1 L3 Cache PMC Events****L3PMCx04 [L3 tag lookup state] (Core::X86::Pmc::L3::L3LookupState)**

Read-write.

All L3 Requests.

L3PMCx04

Bits	Description												
7:0	<b>L3LookupMask</b> . Read-write. L3 Request Types												
<b>ValidValues:</b>													
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>01h</td><td>L3 Miss</td></tr> <tr> <td>FDh-02h</td><td>Reserved.</td></tr> <tr> <td>FEh</td><td>L3 Hit</td></tr> <tr> <td>FFh</td><td>All coherent accesses to L3</td></tr> </table>	Value	Description	00h	Reserved.	01h	L3 Miss	FDh-02h	Reserved.	FEh	L3 Hit	FFh	All coherent accesses to L3
Value	Description												
00h	Reserved.												
01h	L3 Miss												
FDh-02h	Reserved.												
FEh	L3 Hit												
FFh	All coherent accesses to L3												

**L3PMCxAC [L3 XiSampledLatency] (Core::X86::Pmc::L3::L3\_XiSampledLatency)**

Read-write.

When used in conjunction with L3\_XiSampledLatencyRequests, this PMC Event will measure the average memory latency (excluding MMIO) observed by this CCX.

Configure two PMCs with the L3\_XiSampledLatency and L3\_XiSampledLatencyRequests events and use the following equation to identify the observed latency.

$$\text{Average Sampled Latency} = \text{L3\_XiSampledLatency} / \text{L3\_XiSampledLatencyRequests} * 10\text{ns}$$

Some ChL3PmcCfg fields must be programmed as follows to ensure that these events accurately measure latency: ChL3PmcCfg[EnAllSources]=0x1.

Other ChL3PmcCfg fields can be used to filter the measured latency based on originating thread (EnAllCores, CoreID) and Data Source (UnitMask).

To measure average latency from all threads to all Data Sources, use the following configuration:

ChL3PmcCfg[EnAllCores]=0x1, ChL3PmcCfg[ThreadMask]=0x3, and ChL3PmcCfg[UnitMask]=0xFF.

L3PMCxAC

Bits	Description
7:6	Reserved.
5	<b>Ext_Far.</b> Read-write. Requests that target another NUMA node and return from Extension Memory (CXL™)
4	<b>Ext_Near.</b> Read-write. Requests that target the same NUMA node and return from Extension Memory (CXL)
3	<b>NearCache_FarCache_Far.</b> Read-write. Requests that target another NUMA node and return from another CCX's cache.
2	<b>NearCache_FarCache_Near.</b> Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	<b>Dram_Far.</b> Read-write. Requests that target another NUMA node and return from DRAM
0	<b>Dram_Near.</b> Read-write. Requests that target the same NUMA node and return from DRAM

**L3PMCxAD [L3 XiSampledLatencyRequests] (Core::X86::Pmc::L3::L3\_XiSampledLatencyRequests)**

Read-write.

When used in conjunction with L3\_XiSampledLatency, this PMC Event will measure the average memory latency (excluding MMIO) observed by this CCX.

Configure two PMCs with the L3\_XiSampledLatency and L3\_XiSampledLatencyRequests events and use the following equation to identify the observed latency.

$$\text{Average Sampled Latency} = \text{L3\_XiSampledLatency} / \text{L3\_XiSampledLatencyRequests} * 10\text{ns}$$

Some ChL3PmcCfg fields must be programmed as follows to ensure that these events accurately measure latency: ChL3PmcCfg[EnAllSources]=0x1.

Other ChL3PmcCfg fields can be used to filter the measured latency based on originating thread (EnAllCores, CoreID) and Data Source (UnitMask).

To measure average latency from all threads to all Data Sources, use the following configuration:

ChL3PmcCfg[EnAllCores]=0x1, ChL3PmcCfg[ThreadMask]=0x3, and ChL3PmcCfg[UnitMask]=0xFF.

L3PMCxAD

Bits	Description
7:6	Reserved.
5	<b>Ext_Far.</b> Read-write. Requests that target another NUMA node and return from Extension Memory (CXL)
4	<b>Ext_Near.</b> Read-write. Requests that target the same NUMA node and return from Extension Memory (CXL)
3	<b>NearCache_FarCache_Far.</b> Read-write. Requests that target another NUMA node and return from another CCX's cache.
2	<b>NearCache_FarCache_Near.</b> Read-write. Requests that target the same NUMA node and return from another CCX's cache.
1	<b>Dram_Far.</b> Read-write. Requests that target another NUMA node and return from DRAM
0	<b>Dram_Near.</b> Read-write. Requests that target the same NUMA node and return from DRAM

### 2.1.15 Instruction Based Sampling (IBS)

IBS is a code profiling mechanism that enables the processor to select a random instruction fetch or macro-op after a programmed time interval has expired and record specific performance information about the operation. An interrupt is generated when the operation is complete as specified by Core::X86::Msrr::IBS\_CTL. An interrupt handler can then read the performance information that was logged for the operation.

The IBS mechanism is split into two parts: instruction fetch performance controlled by Core::X86::Msrr::IBS\_FETCH\_CTL; and instruction execution performance controlled by Core::X86::Msrr::IBS\_OP\_CTL. Instruction fetch sampling provides information about instruction TLB and instruction cache behavior for fetched instructions. Instruction execution sampling provides information about op execution behavior. The data collected for instruction fetch performance is independent from the data collected for instruction execution performance. Support for the IBS feature is indicated by the Core::X86::Cpuid::FeatureExtIdEcX[IBS].

Instruction fetch performance is profiled by recording the following performance information for the tagged instruction fetch:

- If the instruction fetch completed or was aborted. See Core::X86::Msrr::IBS\_FETCH\_CTL.
- The number of clock cycles spent on the instruction fetch. See Core::X86::Msrr::IBS\_FETCH\_CTL.
- If the instruction fetch hit or missed the IC, hit/missed in the L1 and L2 TLBs, and page size. See Core::X86::Msrr::IBS\_FETCH\_CTL.
- The linear address, physical address associated with the fetch. See Core::X86::Msrr::IBS\_FETCH\_LINADDR, Core::X86::Msrr::IBS\_FETCH\_PHYSADDR.

Instruction execution performance is profiled by tagging one macro-op associated with an instruction. Instructions that decode to more than one macro-op return different performance data depending upon which macro-op associated with the instruction is tagged. These macro-ops are associated with the RIP of the next instruction to retire. The following performance information is returned for the tagged op:

- Branch and execution status. See Core::X86::Msrb::IBS\_OP\_DATA.
- Branch target address for branch ops. See Core::X86::Msrb::BP\_IBSTGT\_RIP.
- The logical address associated with the op. See Core::X86::Msrb::IBS\_OP\_RIP.
- The linear and physical address associated with a load or store op. See Core::X86::Msrb::IBS\_DC\_LINADDR, Core::X86::Msrb::IBS\_DC\_PHYSADDR.
- The data cache access status associated with the op: DC hit/miss, DC miss latency, TLB hit/miss, TLB page size. See Core::X86::Msrb::IBS\_OP\_DATA3.
- The number clocks from when the op was tagged until the op retires. See Core::X86::Msrb::IBS\_OP\_DATA.
- The number clocks from when the op completes execution until the op retires. See Core::X86::Msrb::IBS\_OP\_DATA.
- Source information for DRAM and MMIO. See Core::X86::Msrb::IBS\_OP\_DATA2.

## 2.2 L3 Cache

The Level-3 cache (L3) forms the third level of cache in the CPU caching hierarchy. The L3 is a shared, unified cache inside a core complex.

### 2.2.1 L3 MSR Registers

MSR0000_0C81 [L3 QoS Configuration] (L3::L3CRB::L3QosCfg1)	
Reset: 0000_0000_0000_0000h.	
QOS L3 Cache Allocation CDP mode enable (I vs. D). Contents are copied to ChL2QosCfg1 and ChL3QosCfg1_0.	
_ccd[1:0]_lthree0; MSR0000_0C81	
Bits	Description
63:1	Reserved.
0	<b>CDP.</b> Read-write. Reset: 0. Code and Data Prioritization Technology enable
MSR0000_0C90 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask0)	
Reset: 0000_0000_0000_FFFFh.	
QOS L3 Allocation Mask for CLOS0	
_ccd[1:0]_lthree0; MSR0000_0C90	
Bits	Description
63:16	Reserved.
15:0	<b>WayMask.</b> Read-write. Reset: FFFFh. L3 way mask used for allocation control.
MSR0000_0C91 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask1)	
Reset: 0000_0000_0000_FFFFh.	
QOS L3 Allocation Mask for CLOS1	
_ccd[1:0]_lthree0; MSR0000_0C91	
Bits	Description
63:16	Reserved.
15:0	<b>WayMask.</b> Read-write. Reset: FFFFh. L3 way mask used for allocation control.

**MSR0000\_0C92 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask2)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS2

\_ccd[1:0]\_lthree0; MSR0000\_0C92

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C93 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask3)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS3

\_ccd[1:0]\_lthree0; MSR0000\_0C93

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C94 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask4)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS4

\_ccd[1:0]\_lthree0; MSR0000\_0C94

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C95 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask5)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS5

\_ccd[1:0]\_lthree0; MSR0000\_0C95

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C96 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask6)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS6

\_ccd[1:0]\_lthree0; MSR0000\_0C96

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C97 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask7)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS7

\_ccd[1:0]\_lthree0; MSR0000\_0C97

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.

**MSR0000\_0C98 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask8)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS8

\_ccd[1:0]\_lthree0; MSR0000\_0C98

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C99 [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask9)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS9

\_ccd[1:0]\_lthree0; MSR0000\_0C99

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C9A [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask10)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS10

\_ccd[1:0]\_lthree0; MSR0000\_0C9A

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C9B [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask11)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS11

\_ccd[1:0]\_lthree0; MSR0000\_0C9B

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C9C [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask12)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS12

\_ccd[1:0]\_lthree0; MSR0000\_0C9C

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C9D [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask13)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS13

\_ccd[1:0]\_lthree0; MSR0000\_0C9D

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.

**MSR0000\_0C9E [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask14)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS14

\_ccd[1:0]\_lthree0; MSR0000\_0C9E

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSR0000\_0C9F [L3 QoS Allocation Mask] (L3::L3CRB::L3QosAllocMask15)**

Reset: 0000\_0000\_0000\_FFFFh.

QOS L3 Allocation Mask for CLOS15

\_ccd[1:0]\_lthree0; MSR0000\_0C9F

**Bits Description**

63:16 Reserved.

15:0 **WayMask.** Read-write. Reset: FFFFh. L3 way mask used for allocation control.**MSRC000\_0200 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl0)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control0

\_ccd[1:0]\_lthree0; MSRC000\_0200

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0201 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl1)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control1

\_ccd[1:0]\_lthree0; MSRC000\_0201

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0202 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl2)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control2

\_ccd[1:0]\_lthree0; MSRC000\_0202

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0203 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl3)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control3

\_ccd[1:0]\_lthree0; MSRC000\_0203

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value

**MSRC000\_0204 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl4)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control4

\_ccd[1:0]\_lthree0; MSRC000\_0204

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0205 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl5)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control5

\_ccd[1:0]\_lthree0; MSRC000\_0205

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0206 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl6)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control6

\_ccd[1:0]\_lthree0; MSRC000\_0206

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0207 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl7)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control7

\_ccd[1:0]\_lthree0; MSRC000\_0207

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0208 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl8)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control8

\_ccd[1:0]\_lthree0; MSRC000\_0208

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0209 [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl9)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control9

\_ccd[1:0]\_lthree0; MSRC000\_0209

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value



**MSRC000\_020A [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl10)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control10

\_ccd[1:0]\_lthree0; MSRC000\_020A

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_020B [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl11)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control11

\_ccd[1:0]\_lthree0; MSRC000\_020B

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_020C [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl12)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control12

\_ccd[1:0]\_lthree0; MSRC000\_020C

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_020D [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl13)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control13

\_ccd[1:0]\_lthree0; MSRC000\_020D

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_020E [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl14)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control14

\_ccd[1:0]\_lthree0; MSRC000\_020E

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_020F [L3 QoS Bandwidth Control] (L3::L3CRB::L3QosBwControl15)**

Reset: 0000\_0000\_0000\_1000h.

QOS BW Control15

\_ccd[1:0]\_lthree0; MSRC000\_020F

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value

**MSRC000\_0280 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_0)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control0

\_ccd[1:0]\_lthree0; MSRC000\_0280

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0281 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_1)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control1

\_ccd[1:0]\_lthree0; MSRC000\_0281

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0282 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_2)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control2

\_ccd[1:0]\_lthree0; MSRC000\_0282

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0283 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_3)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control3

\_ccd[1:0]\_lthree0; MSRC000\_0283

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0284 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_4)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control4

\_ccd[1:0]\_lthree0; MSRC000\_0284

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0285 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_5)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control5

\_ccd[1:0]\_lthree0; MSRC000\_0285

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value

**MSRC000\_0286 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_6)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control6

\_ccd[1:0]\_lthree0; MSRC000\_0286

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0287 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_7)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control7

\_ccd[1:0]\_lthree0; MSRC000\_0287

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0288 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_8)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control8

\_ccd[1:0]\_lthree0; MSRC000\_0288

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_0289 (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_9)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control9

\_ccd[1:0]\_lthree0; MSRC000\_0289

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_028A (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_10)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control10

\_ccd[1:0]\_lthree0; MSRC000\_028A

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_028B (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_11)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control11

\_ccd[1:0]\_lthree0; MSRC000\_028B

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value

**MSRC000\_028C (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_12)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control12

\_ccd[1:0]\_lthree0; MSRC000\_028C

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_028D (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_13)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control13

\_ccd[1:0]\_lthree0; MSRC000\_028D

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_028E (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_14)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control14

\_ccd[1:0]\_lthree0; MSRC000\_028E

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value**MSRC000\_028F (L3::L3CRB::L3QOS\_SLOWBW\_CONTROL\_15)**

Reset: 0000\_0000\_0000\_1000h.

QOS Slow Memory BW Control15

\_ccd[1:0]\_lthree0; MSRC000\_028F

**Bits Description**

63:13 Reserved.

12:0 **Ceiling.** Read-write. Reset: 1000h. QOS BW Control BW ceiling value

**MSRC000\_03FD (L3::L3CRB::L3\_QOS\_ABMC\_CFG)**

Reset: 0000\_0000\_0000\_007Fh.

L3 QOS ABMC Counter Config Register.

\_ccd[1:0]\_lthree0; MSRC000\_03FD

Bits	Description
63	<b>ConfigureCounter.</b> Read-write. Reset: 0. Configure the specified counter. Packed from MSR[63]
62	<b>EnableCounter.</b> Read-write. Reset: 0. Enable the specified counter. Packed from MSR[62]
61:53	Reserved.
52:48	<b>CounterId.</b> Read-write. Reset: 00h. ID of the counter to configure (or describe). Packed from MSR[52:48]
47	<b>BwSrcIsClos.</b> Read-write. Reset: 0. 0=RMID, 1=CLOS. Identifies if the BwSrc identifies an RMID or a CLOS. Packed from MSR[47]
46:44	Reserved.
43:32	<b>BwSrc.</b> Read-write. Reset: 000h. Which RMID (or CLOS) to track with the counter. Packed from MSR[43:32]
31:7	Reserved.
6	<b>L3CacheVicBwMon.</b> Read-write. Reset: 1. Dirty Victims from the QOS domain to all types of memory
5	<b>L3CacheRmtSlowBwFillMon.</b> Read-write. Reset: 1. Reads to slow memory in the non-local NUMA domain
4	<b>L3CacheLclSlowBwFillMon.</b> Read-write. Reset: 1. Reads to slow memory in the local NUMA domain
3	<b>L3CacheRmtBwNtWrMon.</b> Read-write. Reset: 1. Non-temporal writes to non-local NUMA domain
2	<b>L3CacheLclBwNtWrMon.</b> Read-write. Reset: 1. Non-temporal writes to local NUMA domain
1	<b>L3CacheRmtBwFillMon.</b> Read-write. Reset: 1. Reads to memory in the non-local NUMA domain
0	<b>L3CacheLclBwFillMon.</b> Read-write. Reset: 1. Reads to memory in the local NUMA domain

**MSRC000\_03FE (L3::L3CRB::L3\_QOS\_ABMC\_DSC)**

Read-only, Volatile. Reset: 0000\_0000\_0000\_007Fh.

L3 QOS ABMC Counter Config Register.

\_ccd[1:0]\_lthree0; MSRC000\_03FE

Bits	Description
63	<b>ConfigurationError.</b> Read-only, Volatile. Reset: 0. Configure the specified counter. Packed from MSR[63]
62	<b>EnableCounter.</b> Read-only, Volatile. Reset: 0. Enable the specified counter. Packed from MSR[62]
61:53	Reserved.
52:48	<b>CounterId.</b> Read-only, Volatile. Reset: 00h. ID of the counter to configure (or describe). Packed from MSR[52:48]
47	<b>BwSrcIsClos.</b> Read-only, Volatile. Reset: 0. 0=RMID, 1=CLOS. Identifies if the BwSrc identifies an RMID or a CLOS. Packed from MSR[47]
46:44	Reserved.
43:32	<b>BwSrc.</b> Read-only, Volatile. Reset: 000h. Which RMID (or CLOS) to track with the counter. Packed from MSR[43:32]
31:7	Reserved.
6	<b>L3CacheVicBwMon.</b> Read-only, Volatile. Reset: 1. Dirty Victims from the QOS domain to all types of memory
5	<b>L3CacheRmtSlowBwFillMon.</b> Read-only, Volatile. Reset: 1. Reads to slow memory in the non-local NUMA domain
4	<b>L3CacheLclSlowBwFillMon.</b> Read-only, Volatile. Reset: 1. Reads to slow memory in the local NUMA domain
3	<b>L3CacheRmtBwNtWrMon.</b> Read-only, Volatile. Reset: 1. Non-temporal writes to non-local NUMA domain
2	<b>L3CacheLclBwNtWrMon.</b> Read-only, Volatile. Reset: 1. Non-temporal writes to local NUMA domain
1	<b>L3CacheRmtBwFillMon.</b> Read-only, Volatile. Reset: 1. Reads to memory in the non-local NUMA domain
0	<b>L3CacheLclBwFillMon.</b> Read-only, Volatile. Reset: 1. Reads to memory in the local NUMA domain

**MSRC000\_03FF [L3 Qos Extended Configuration] (L3::L3CRB::L3QosExtCfg)**

Reset: 0000\_0000\_0000\_0000h.

AMD specific register for BW control (not x86 ISA specified)

\_ccd[1:0]\_lthree0; MSRC000\_03FF

Bits	Description
63:2	Reserved.
1	<b>CdmaToMaxCbm_En.</b> Read-write. Reset: 0. Enable ABMC
0	<b>ABMC_En.</b> Read-write. Reset: 0. Enable ABMC

**MSRC000\_0400 (L3::L3CRB::QOS\_EVT\_CFG\_0)**

Reset: 0000\_0000\_0000\_007Fh.

Identifies the Bandwidth Sources to include in Bandwidth Monitoring Event 0

\_ccd[1:0]\_lthree0; MSRC000\_0400

Bits	Description
63:7	Reserved.
6	<b>L3CacheVicBwMon.</b> Read-write. Reset: 1. Dirty Victims from the QOS domain to all types of memory
5	<b>L3CacheRmtSlowBwFillMon.</b> Read-write. Reset: 1. Reads to slow memory in the non-local NUMA domain
4	<b>L3CacheLclSlowBwFillMon.</b> Read-write. Reset: 1. Reads to slow memory in the local NUMA domain
3	<b>L3CacheRmtBwNtWrMon.</b> Read-write. Reset: 1. Non-temporal writes to non-local NUMA domain
2	<b>L3CacheLclBwNtWrMon.</b> Read-write. Reset: 1. Non-temporal writes to local NUMA domain
1	<b>L3CacheRmtBwFillMon.</b> Read-write. Reset: 1. Reads to memory in the non-local NUMA domain
0	<b>L3CacheLclBwFillMon.</b> Read-write. Reset: 1. Reads to memory in the local NUMA domain

**MSRC000\_0401 (L3::L3CRB::QOS\_EVT\_CFG\_1)**

Reset: 0000\_0000\_0000\_0015h.

Identifies the Bandwidth Sources to include in Bandwidth Monitoring Event 1

\_ccd[1:0]\_lthree0; MSRC000\_0401

Bits	Description
63:7	Reserved.
6	<b>L3CacheVicBwMon.</b> Read-write. Reset: 0. Dirty Victims from the QOS domain to all types of memory
5	<b>L3CacheRmtSlowBwFillMon.</b> Read-write. Reset: 0. Reads to slow memory in the non-local NUMA domain
4	<b>L3CacheLclSlowBwFillMon.</b> Read-write. Reset: 1. Reads to slow memory in the local NUMA domain
3	<b>L3CacheRmtBwNtWrMon.</b> Read-write. Reset: 0. Non-temporal writes to non-local NUMA domain
2	<b>L3CacheLclBwNtWrMon.</b> Read-write. Reset: 1. Non-temporal writes to local NUMA domain
1	<b>L3CacheRmtBwFillMon.</b> Read-write. Reset: 0. Reads to memory in the non-local NUMA domain
0	<b>L3CacheLclBwFillMon.</b> Read-write. Reset: 1. Reads to memory in the local NUMA domain

**MSRC001\_023[1...B] [L3 Performance Monitor Counter] (L3::L3CRB::ChL3Pmc)**

Read-write, Volatile. Reset: 0000\_0000\_0000\_0000h.

L3 Performance Monitor Counter Register

\_ccd[1:0]\_lthree0\_n0; MSRC001\_0231

\_ccd[1:0]\_lthree0\_n1; MSRC001\_0233

\_ccd[1:0]\_lthree0\_n2; MSRC001\_0235

\_ccd[1:0]\_lthree0\_n3; MSRC001\_0237

\_ccd[1:0]\_lthree0\_n4; MSRC001\_0239

\_ccd[1:0]\_lthree0\_n5; MSRC001\_023B

\_ccd0\_lthree0\_n[5:0]\_aliasSMN; L3L3CRBx2037\_80[30,28,20,18,10,08]; L3L3CRB=0000\_0000h

\_ccd1\_lthree0\_n[5:0]\_aliasSMN; L3L3CRBx20B7\_80[30,28,20,18,10,08]; L3L3CRB=0000\_0000h

Bits	Description
63:49	Reserved.
48	<b>Overflow.</b> Read-write, Volatile. Reset: 0. Counter overflow bit
47:32	<b>CountHi.</b> Read-write, Volatile. Reset: 0000h. Bits 47:32 of the count
31:0	<b>CountLo.</b> Read-write, Volatile. Reset: 0000_0000h. Bits 31:0 of the count

**MSRC001\_1095 [L3 Cache Range Reserve Base Address] (L3::L3CRB::L3RangeReserveBaseAddr)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0; MSRC001\_1095

Bits	Description
63:52	Reserved.
51:12	<b>Addr.</b> Read-write. Reset: 00_0000_0000h. Base physical address bits 51:12 for the locked range.
11:0	Reserved.

**MSRC001\_1096 [L3 Cache Range Reserve Maximum Address] (L3::L3CRB::L3RangeReserveMaxAddr)**

Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_lthree0; MSRC001\_1096

Bits	Description
63:52	Reserved.
51:12	<b>Addr.</b> Read-write. Reset: 00_0000_0000h. Max physical address bits 51:12 for the locked range.
11:1	Reserved.
0	<b>En.</b> Read-write. Reset: 0. 0=Disable L3 Range Reservation. 1=Enable L3 Range Reservation. Enables the L3 range reservation when set.

**MSRC001\_109A [L3 Cache Range Reservation Way Mask] (L3::L3CRB::L3RangeReserveWayMask)**

Reset: 0000\_0000\_0000\_0000h.

Way mask used to specify which L3 cache ways are used for range reservation

\_ccd[1:0]\_lthree0; MSRC001\_109A

Bits	Description
63:16	Reserved.
15:0	<b>Mask.</b> Read-write. Reset: 0000h. L3 ways used for range reservation.

**2.2.2 L3 Clocks and Test (CT) MSR Registers.**

**MSRC001\_0299 (L3::L3CT::L3RAPLPowerUnit0)**

Read-only. Reset: 0000\_0000\_0000\_0000h.

L3 RAPL Power Unit 0

\_ccd[1:0]\_lthree0; MSRC001\_0299

Bits	Description
63:20	Reserved.
19:16	<b>TimeUnits.</b> Read-only. Reset: 0h. Time information (in Seconds) is based on the multiplier, $1/2^{TU}$ where TU is unsigned. Default value is 1010b, indicating time unit is in 976 microseconds increment
15:13	Reserved.
12:8	<b>EnergyStatusUnits.</b> Read-only. Reset: 00h. Energy information (in Joules) is based on the multiplier, $1/2^{ESU}$ where ESU is unsigned integer. Default value is 10000b, indicating energy status unit is in 15.3 micro-Joules increment
7:4	Reserved.
3:0	<b>PowerUnits.</b> Read-only. Reset: 0h. Power information (in Watts) is based on the multiplier, $1/2^{PU}$ where PU is an unsigned integer. Default value is 0011b, indicating power unit is in 1/8 Watts increment

**MSRC001\_029B (L3::L3CT::L3CCXEnergyStatus)**

Read-only. Reset: 0000\_0000\_0000\_0000h.

L3 CCX Energy Status 0

\_ccd[1:0]\_lthree0; MSRC001\_029B

Bits	Description
63:0	<p><b>TotalEnergyConsumed.</b> Read-only. Reset: 0000_0000_0000_0000h.</p> <p><b>Description:</b> Total Energy consumed since the last time the register is cleared. It reports the actual energy use for respective power domain. This MSR is updated every ~1ms. Energy status is free running. Users calculate power for a given domain by calculating <math>dEnergy/dTime</math> for that domain. Users must ensure successive reads contain atleast one, but preferably many energy status updates by hardware. Readable/writable field for use by SMU.</p>



### 3 Reliability, Availability, and Serviceability (RAS) Features

A full implementation of RAS involves capabilities and support from the processor design, board hardware design, BIOS, firmware, and software.

#### 3.1 Machine Check Architecture

*Table 21: Machine Check Terms and Acronyms*

Term	Description
<b>MCA</b>	Machine Check Architecture.
<b>MCAX</b>	Machine Check Architecture eXtensions.
<b>WRIG</b>	Writes Ignored.

##### 3.1.1 Overview

The processor contains logic and registers to detect, log, and correct errors in the data or control paths. The Machine Check Architecture (MCA) defines facilities by which processor and system hardware errors are logged and reported to system software. This allows system software to perform a strategic role in recovery from and diagnosis of hardware errors.

##### 3.1.1.1 Legacy Machine Check Architecture

The legacy x86 Machine Check Architecture (MCA) refers to the standard x86 facilities for error logging and reporting. Refer to the AMD64 Architecture Programmer's Manual for an architectural overview of the Machine Check Architecture.

Support for the MCA is indicated by Core::X86::Cpuid::FeatureIdEdx[MCA] or Core::X86::Cpuid::FeatureExtIdEdx[MCA].

##### 3.1.1.2 Machine Check Architecture Extensions

Machine Check Architecture Extensions (MCAX) is AMD's x86-64 extension to the Machine Check Architecture.

Goals of MCAX include:

- Accommodate a variety of implementations, where each implementation may have a different assignment of MCA bank to block.
  - For example, one implementation may have 1 memory channel with an MCA bank, and another otherwise identical implementation may have 2 memory channels, each with their own MCA bank. Therefore, MCA bank allocation will appear different between these two implementations. MCAX is designed to require no assumptions about which MCA banks access which blocks.
  - Provide granular information for error logging, to improve error handling and diagnosability.
  - Preserve compatibility with system software which is not MCAX-aware.

Features of the MCA Extensions include:

- Increased MCA Bank Count: Features to support an expansion of the number of MCA banks supported by AMD processors.
- MCA Extension Registers: Expanded information logged in MCA banks to allow for improved error handling, better diagnosability, and future scalability.
- MCA DOER/SEER Roles: Separation of MCA information to take advantage of emerging software roles, namely

Error Management (Dynamic Operational Error Handling, or DOER) for managing running programs, and Fault Management (Symptom Elaboration of Errors, or SEER) for hardware diagnosability and reconfiguration. This clearer separation is accompanied by the assurances of architectural state (vs. implementation dependent state), so that operating systems can rely on the state and exploit new functionality.

Support for Machine Check Architecture Extensions (MCAX) is indicated by `Core::X86::Cpuid::RasCap[ScalableMca]`.

### 3.1.1.3 Use of MCA Information

The MCA registers contain information that can be used for multiple purposes. Some of this information is architecturally specified, and remains consistent from generation to generation, enabling portable, stable code. Some of this information is implementation specific; it is vital for diagnosis and other software functions, but may change with new implementations. It is important to understand how this information is categorized, and how it should be used. This section describes a framework for that.

There are two fundamental roles to be carried out after an error occurs; Error Management and Fault Management. All information required for Error Management is architectural and stable; some information required for Fault Management is also architectural.

#### 3.1.1.3.1 Error Management

Error Management describes actions necessary by operational software (e.g., the operating system or the hypervisor) to manage running programs that are affected by the error. The list of possible actions for operational error management is generally fairly short: take no action; terminate a single affected process, program, or virtual machine; terminate system operation. The Error Management role is defined as the DOER role (Dynamic Operational Error Handling). The name is intended to indicate an active role in managing running programs. Information used by the DOER is fairly limited and straightforward. It includes only those status fields needed to make decisions about the scope and severity of the error, and to determine what immediate action is to be taken.

#### 3.1.1.3.2 Fault Management

Fault Management describes optional actions for purposes of diagnosis, repair, and reconfiguration of the underlying hardware. The Fault Management role is described as SEER (Symptom Elaboration of Errors) because it peers further into hardware behavior and may try to influence future behavior via Predictive Fault Analysis, reconfiguration, service actions, etc. Because the SEER depends on understanding specifics of hardware configuration, it necessarily requires implementation specific knowledge and may not be portable across implementations.

Fields that are not explicitly specified as DOER are SEER. By separating error handling software into DOER and SEER roles, programmers can create both simpler and more functional code. The terms DOER and SEER appear in other sections of this document as an aid to reasoning about error handling and understanding actions to be taken.

### 3.1.2 Machine Check Registers

Host software references MCA registers via MSRs. MSRs are accessed through x86 WRMSR and RDMSR instructions. MSR addresses are private to a logical core; a given MSR referenced by two different cores results in references to two different MCA registers.

#### 3.1.2.1 Global Registers

`Core::X86::Cpuid::FeatureIdEdx[MCA]` or `Core::X86::Cpuid::FeatureExtIdEdx[MCA]` indicates the presence of the following machine check registers:

- Core::X86::Msr::MCG\_CAP
  - Reports how many machine check register banks are supported. This value reflects the number of MCA banks visible to that logical core. Some banks may be RAZ/WRIG either due to the bank being reserved or unused on this processor or because the block's MCA bank is controlled by another logical core.
- Core::X86::Msr::MCG\_STAT
  - Provides basic information about processor state after the occurrence of a machine check error.
- Core::X86::Msr::MCG\_CTL
  - Used by software to enable or disable the logging and reporting of machine check errors in the error reporting banks. Some bits may be RAZ/WRIG either due to the bank being reserved or unused on this processor or because the block's MCA bank is controlled by another logical core.
- Core::X86::Msr::McaIntrCfg
  - Used by software to configure certain machine check interrupts.

### 3.1.2.2 Machine Check Banks

A processor contains multiple blocks, and some of them have banks of machine check architecture registers (MCA banks). An MCA bank logs and reports errors to software.

The legacy MCA supports up to 32 MCA banks per logical core. MCAX supports up to 64 MCA banks per logical core.

The processor ensures that non-zero error status in an MCA bank is visible to exactly one logical core in a system, and that error notifications are directed to that logical core. Hardware also makes MCA bank configuration and control registers available to exactly one logical core. Banks associated with a CPU core are controlled by that logical core. Banks associated with other blocks are controlled by an implementation-specific logical core.

#### 3.1.2.2.1 Legacy MCA Registers

Each legacy MCA bank allocates address space for 4 legacy MCA registers.

The legacy MCA registers include:

- MCA\_CTL
  - Enables error reporting via machine check exception.
- MCA\_STATUS
  - Logs information associated with errors.
- MCA\_ADDR
  - Logs address information associated with errors.
- MCA\_MISC0
  - Logs miscellaneous information associated with errors.

#### 3.1.2.2.2 Legacy MCA MSRs

The legacy MCA MSRs are MSR0000\_04[7F:00]. The legacy MCA MSR space contains 32 banks of 4 registers per bank. The layout of the legacy MCA MSR space is given in Table 22 [Legacy MCA MSR Layout].

Table 22: Legacy MCA MSR Layout

MCA bank (decimal)	MCA_CTL (MSR0000_0xxx)	MCA_STATUS	MCA_ADDR	MCA_MISC0
0	400	401	402	403
1	404	405	406	407
2	408	409	40A	40B
3	40C	40D	40E	40F

4	410	411	412	413
5	414	415	416	417
6	418	419	41A	41B
...				
31	47C	47D	47E	47F

Features and registers associated with the MCA Extensions are not available in this legacy MSR address range. AMD recommends that operating systems use the MCAX MSR address range, rather than rely on the legacy MCA MSR address range.

All unimplemented or unused registers in the legacy MCA MSR address range are RAZ/WRIG. MC4 registers (MSR0000\_0410:0000\_0413) are RAZ/WRIG.

MSR0000\_0000 is aliased to the MCAX MSR address for MC0\_ADDR, and MSR0000\_0001 is aliased to the MCAX MSR address of MC0\_STATUS.

### 3.1.2.2.3 MCAX Registers

Each MCAX bank allocates address space for 16 MCA registers. All unimplemented registers in the MCA MSR space are RAZ/WRIG. MCAX bank registers include the legacy MCA registers as well as registers associated with the MCA Extensions.

The MCA Extension registers include:

- MCA\_CONFIG
  - Provide configuration capabilities for this MCA bank.
- MCA\_IPID
  - Provides information on the block associated with this MCA bank.
- MCA\_SYND
  - Logs physical location information associated with a logged error.
- MCA\_DESTATUS
  - Logs status information associated with a deferred error.
- MCA\_DEADDR
  - Logs address information associated with a deferred error.
- MCA\_MISC[1:4]
  - Provides additional threshold counters within an MCA bank.
- MCA\_SYND1 & MCA\_SYND2
  - Log information associated with a logged error, such as FruText.

### 3.1.2.2.4 MCAX MSRs

MCAX MSRs are present at MSRC000\_2[3FF:000]. This MSR address range contains space for 64 banks of 16 registers each. MSRC000\_2[FFF:400] are Reserved for future use. The MCAX MSR address range allows access to both legacy MCA registers and MCAX registers in each MCA bank.

The x86 MCAX MSR address format is SSSS\_SBBR (hex). S=MCA register space (i.e., MSRC000\_2XXX). B=MCA bank. R=Register offset within MCA bank. The layout of the MCAX MSR space is given in Table 23 [MCAX MSR Layout].

Access to unused MCAX MSRs is RAZ/WRIG. MCA Bank 4 is always Read-as-zero (RAZ/WRIG).

Table 23: MCAX MSR Layout

MCA bank	MCAX MSR (MSRC000_2xxx)												
	Legacy MCA Bank registers				MCAX Bank registers								
	CTL	STATUS	ADDR	MISC0	CONFIG	IPID	SYND	Reserved	DESTAT	DEADDR	MISC[4:1]	SYND1	SYND2
0	000	001	002	003	004	005	006	007	008	009	00D:00A	00E	00F
1	010	011	012	013	014	015	016	017	018	019	01D:01A	01E	01F
2	020	021	022	023	024	025	026	027	028	029	02D:02A	02E	02F
...													
63	3F0	3F1	3F2	3F3	3F4	3F5	3F6	3F7	3F8	3F9	3FD:3FA	3FE	3FF

All processors maintain the same mapping of MSR to MCA bank number (MSRC000\_2000 for the beginning of MCA Bank 0, MSRC000\_2010 for the beginning of MCA Bank 1, etc.), regardless of what block the bank represents (see 3.1.5.5 [Determining Bank Type]).

MCA\_CTL\_MASK MSRs are present at MSRC001\_04[3F:00]. MSRC001\_04[FF:40] are Reserved for future use. The layout of these registers is given in Table 24 [MCAX Implementation-Specific Register Layout].

Table 24: MCAX Implementation-Specific Register Layout

MCA bank	MCA_CTL_MASK (MSRC001_04xx)
0	00
1	01
2	02
...	
63	3F

### 3.1.2.3 Access Permissions

When McStatusWrEn == 0, a Write to an implemented MCA\_STATUS register causes a General Protection Fault (#GP) unless the value being written is zero. When McStatusWrEn == 1, a Write to an implemented MCA\_STATUS register does not cause a #GP regardless of data value.

Access to legacy MCA\_CTL\_MASK (MSRC001\_00xx) causes a General Protection Fault (#GP).

Access to legacy MC4\_MISC1-8 (MSRC000\_0408:C000\_040F) is RAZ/WRIG.

### 3.1.3 Machine Check Errors

#### 3.1.3.1 Error Severities

The classes of machine check errors are, in priority order from highest to lowest:

- Uncorrected
- Deferred
- Corrected

Uncorrected errors cannot be corrected by hardware. Uncorrected errors update the status and address registers if not masked from logging in MCA\_CTL\_MASK. Information in the status and address registers from a previously logged lower priority error is overwritten. Previously logged errors of the same priority are not overwritten. Uncorrected errors that are enabled for reporting in MCA\_CTL result in reporting to software via machine check exceptions. If an uncorrected error is masked from logging, the error is ignored by hardware (exceptions are noted in the register definitions). If an uncorrected error is disabled from reporting, containment of the error and logging/reporting of subsequent errors may be affected. Therefore, enable reporting of unmasked uncorrected errors for normal operation. Disable reporting of uncorrected errors only for debug purposes.

Deferred errors are errors that cannot be corrected by hardware, but do not cause an immediate interruption in program flow, loss of data integrity, or corruption of processor state. These errors indicate that data has been corrupted but not consumed; no exception is generated because the data has not been referenced by a core or an IO link. Hardware writes information to the status and address registers in the corresponding bank that identifies the source of the error if deferred errors are enabled for logging. If there is information in the status and address registers from a previously logged lower priority error, it is overwritten. Previously logged errors of the same or higher priority are not overwritten. Deferred errors are not reported via machine check exceptions; they can optionally be reported via LVT or SMI.

Corrected errors are those which have been corrected by hardware and cause no loss of data or corruption of processor state. Hardware writes the status and address registers in the corresponding register bank with information that identifies the source of the error if they are enabled for logging. Corrected errors are not reported via machine check exceptions. Some corrected errors may optionally be reported to software via LVT or SMI if the number of errors exceeds a configurable threshold.

An error to be logged when the status register contains valid data can result in an overflow condition. During error overflow conditions, the new error may not be logged or an error which has already been logged in the status register may be overwritten.

Table 25 [Error Overwrite Priorities] indicates which errors are overwritten in the error status registers.

*Table 25: Error Overwrite Priorities*

		Older Error		
		Uncorrected	Deferred	Corrected
Newer	Uncorrected	-	Overwrite	Overwrite
	Deferred	-	-	Overwrite

Error	Corrected	-	-	-
-------	-----------	---	---	---

Table 26 [Error Scope Hierarchy] provides a hierarchy of error scopes that determine the potential ability to recover the system based on fields in MCA\_STATUS when MCA\_STATUS[Val] == 1.

*Table 26: Error Scope Hierarchy*

PCC	UC	TCC	Deferred	Comments
1	X	X	X	Uncorrected system fatal error. Action required. A hardware-uncorrected error has corrupted system state. The error is fatal to the system and the system processing must be terminated.
0	1	1	X	Uncorrected thread fatal error. Action required. A hardware-uncorrected error has corrupted state for the process thread executing on the interrupted logical core. State for other process threads is unaffected.
0	1	0	X	Uncorrected recoverable error. Action required. A hardware-uncorrected error has not corrupted state of the process thread. Recovery of the process thread is possible if the uncorrected error is corrected by software.
0	0	0	1	Deferred error. Action optional. A hardware-uncorrected error has been discovered but not yet consumed. Error handling software may attempt to correct this error, or prevent access by processes which map the data, or make the physical resource containing the data inaccessible.
0	0	0	0	Corrected error. Action optional. A hardware-corrected error has been corrected. No action is required by error handling software.

### 3.1.3.2 Exceptions and Interrupts

Some or all errors logged in the MCA may require an interrupt or exception to be signaled.

The processor supports the following x86 interrupt/exception types to be communicated to the x86 core in response to an error:

- Machine Check Exception (MCE)
- System Management Interrupt (SMI)
- APIC based interrupt (LVT)

MCEs can be architecturally precise, context-synchronous, or asynchronous. An MCE that sets Core::X86::Msr::MCG\_STAT[RIPV] = 1 and Core::X86::Msr::MCG\_STAT[EIPV] = 1 is precise and the program can be restarted reliably. Other interrupts are architecturally asynchronous.

The ability of hardware to generate a machine check exception upon an error is indicated by Core::X86::Cpuid::FeatureIdEdx[MCE] or Core::X86::Cpuid::FeatureExtIdEdx[MCE].

### 3.1.3.3 Error Codes

The MCA\_STATUS[ErrorCode] field contains information used to identify the logged error. This section identifies how to decode the ErrorCode field.

*Table 27: Error Code Types*

Error Code	Error Code Type	Description
------------	-----------------	-------------

0000 0000 0001 TTLL	TLB	TT = Transaction Type LL = Cache Level
0000 0001 RRRR TTLL	Memory	RRRR = Memory Transaction Type TT = Transaction Type LL = Cache Level
0000 1XXT RRRR XXLL	Bus	XX = Reserved T = Timeout RRRR = Memory Transaction Type LL = Cache Level
0000 01UU 0000 0000	Internal Unclassified	UU = Internal Error Type

Table 28: Error code: transaction type (TT)

TT	Transaction Type
00	Instruction
01	Data
10	Generic
11	Reserved

Table 29: Error codes: cache level (LL)

LL	Cache Level
00	L0: Core
01	L1: Level 1
10	L2: Level 2
11	LG: Generic

Table 30: Error codes: memory transaction type (RRRR)

RRRR	Memory Transaction Type
0000	Generic
0001	Generic Read
0010	Generic Write
0011	Data Read
0100	Data Write
0101	Instruction Fetch
0110	Prefetch
0111	Evict
1000	Snoop (Probe)

Errors can also be identified by the MCA\_STATUS[ErrorCodeExt] field. MCA\_STATUS[ErrorCodeExt] indicates which bit position in the corresponding MCA\_CTL register enables error reporting for the logged error. For instance, MCA\_STATUS[ErrorCodeExt] == 0x9 means that the logged error is enabled by MCA\_CTL[9], and the description of MCA\_CTL[9] contains information on decoding the error log. Specific ErrorCodeExt values are implementation dependent, and should not be used by architectural or portable code.

### 3.1.3.4 Extended Error Codes

The MCA\_STATUS[ErrorCodeExt] field contains additional information used to identify the logged error. Error positions in MCA\_CTL and MCA\_CTL\_MASK and Extended Error Codes are fixed within a given bank type. That is, for an MCA bank with a given MCA\_IPID[HwId, McaType] value, the processor ensures that the same error is reported in a given bit



position of of MCA\_CTL regardless of the product in which that bank appears. Similarly, for an MCA bank with a given MCA\_IPID[HwId, McaType] value, hardware ensures that the mapping of errors to Extended Error Codes is consistent across products.

### 3.1.3.5 DOER and SEER State

The DOER fields are:

- MCG\_STAT
  - Count
  - MCIP
  - RIPV
  - EIPV
- MCA\_STATUS
  - Val
  - PCC
  - TCC
  - UC
  - MiscV
  - AddrV

The MCA\_STATUS[Deferred] bit is used for SEER functionality but is architectural.

### 3.1.3.6 MCA Overflow Recovery

MCA Overflow Recovery is a feature allowing recovery of the system when the overflow bit is set. MCA Overflow Recovery is supported when `Core::X86::Cpuid::RasCap[McaOverflowRecov] == 1`.

When MCA Overflow Recovery is supported, software may rely on `MCA_STATUS[PCC] == 1` to indicate all system-fatal conditions. When MCA Overflow Recovery is not supported, an uncorrected error logged with `MCA_STATUS[Overflow] = 1` may indicate the system-fatal condition that an error requiring software intervention was not logged. Therefore, software must terminate system processing whenever an uncorrected error is logged with `MCA_STATUS[Overflow] = 1`.

### 3.1.3.7 MCA Recovery

MCA Recovery is a feature allowing recovery of the system when the hardware cannot correct an error. MCA Recovery is supported when `Core::X86::Cpuid::RasCap[SUCCOR] == 1`.

When MCA Recovery is supported and an uncorrected error has been detected that the hardware can contain to the task or process to which the machine check has been delivered, it logs a context-synchronous uncorrectable error (`MCA_STATUS[UC] = 1`, `MCA_STATUS[PCC] = 0`). The rest of the system is unaffected and may continue running if supervisory software can terminate only the affected process or VM.

## 3.1.4 Machine Check Features

### 3.1.4.1 Error Thresholding

For some types of errors, the hardware maintains counts of the number of errors. When the counter reaches a programmable threshold, an event may optionally be triggered to signal system software. This is known as error

thresholding. The primary purpose of error thresholding is to help software recognize an excessive rate of errors, which may indicate marginal or failing hardware. This information can be used to make decisions about deconfiguring hardware or scheduling service actions. The error count is incremented for corrected, deferred, and uncorrected errors.

The MCA\_MISCx registers contain the architectural interface for error thresholding. The registers contain a 12-bit error counter that can be initialized to any value except FFFh, with the option to interrupt when the counter reaches FFFh.

MCA\_MISCx[ThresholdIntType] determines the type of interrupt to be generated for threshold overflow errors in that counter. This can be set to None, LVT, or SMI. If this is set to LVT, Core::X86::Ms::McaIntrCfg[ThresholdLvtOffset] specifies the LVT offset that is used. Only one LVT offset is used per socket and the interrupt is routed to the APIC of the logical core from which the MCA bank is visible.

### 3.1.4.2 Error Simulation

Error simulation involves creating the appearance to software that an error occurred, and can be used to debug machine check interrupt handlers. See Core::X86::Ms::HWCR[McStatusWrEn] for making MCA registers writable for non-zero values. When McStatusWrEn is set, privileged software can write non-zero values to the specified registers without generating exceptions, and then simulate a machine check using the INT18 instruction (INTn instruction with an operand of 18). Setting a reserved bit in these registers does not generate an exception when this mode is enabled. However, setting a reserved bit may result in undefined behavior.

### 3.1.5 Software Guidelines

#### 3.1.5.1 Recognizing MCAX Support

Software which reads the MCA registers must recognize whether an implementation uses the legacy format or the MCAX format. This is accomplished by starting with CPUID Fn8000\_0007\_EBX[ScalableMca]. If ScalableMca == 1, then the implementation supports the MCAX indicator (MCA\_CONFIG[Mcac]). An MCA bank is an MCAX bank if MCA\_CONFIG[Mcac] == 1 in that bank.

#### 3.1.5.2 Communicating MCAX Support

Software which supports MCAX must set MCA\_CONFIG[McacEn] = 1 in each MCA bank.

Software that supports MCAX should use the MCAX MSRs to access both legacy and MCAX registers.

#### 3.1.5.3 Machine Check Initialization

The following initialization sequence must be followed:

- Platform firmware must initialize the MCA\_CTL\_MASK registers prior to the initialization of the MCA\_CTL registers and Core::X86::Ms::MCG\_CTL. Platform firmware and the operating system must not clear MCA\_CTL\_MASK bits that are set to 1. MCA\_CTL\_MASK registers must be set the same across all cores.
- The operating system must initialize the MCA\_CONFIG registers prior to initialization of the MCA\_CTL registers.
- The MCA\_CTL registers must be initialized prior to enabling the error reporting banks in MCG\_CTL.
- The Core::X86::Ms::MCG\_CTL register must be programmed identically for all cores in a processor, although the Read-write bits may differ per core.
- CR4.MCE must be set to enable machine check exceptions.

The operating system should configure the MCA\_CONFIG registers as follows:

- MCA\_CONFIG[McaXEn] = 1 if the operating system has been updated to use the MCA Extension MSR addresses. Otherwise, the operating system should preserve the platform firmware-programmed value of this field.
- MCA\_CONFIG[LogDeferredInMcaStat] and MCA\_CONFIG[DeferredIntType] to appropriate values based on OS support for deferred errors.

MCA\_STATUS MSRs are cleared by hardware after a cold reset. If initializing after a warm reset, then platform firmware should check for valid MCA errors and if present save the status for later diagnostic use.

Platform firmware may initialize the MCA without setting CR4.MCE; this results in a shutdown on any machine check which would have caused a machine check exception (followed by a reboot if configured). Alternatively, platform firmware that wishes to ensure continued operation in the event that a machine check occurs during boot may write MCG\_CTL with all ones and write zeros into each MCA\_CTL register. With these settings, a machine check error results in MCA\_STATUS being written without generating a machine check exception or a shutdown. Platform firmware may then poll MCA\_STATUS registers during critical sections of boot to ensure system integrity. Note that the system may be operating with corrupt data before polling MCA\_STATUS registers. Before passing control to the operating system, platform firmware should restore the values of those registers to what the operating system is expecting.

After MCA initialization, system software should check the Val bit on each MCA\_STATUS register. It is possible that valid error status information has already been logged in the MCA\_STATUS registers at the time software is attempting to initialize them. The status can reflect errors logged prior to a warm reset or errors recorded during the system power-up and boot process. Before clearing the MCA\_STATUS registers, software should examine their contents and log any errors found.

#### 3.1.5.4 Determining Bank Count

System software should read Core::X86::Msr::MCG\_CAP[Count] to determine the number of machine check banks visible to a logical core. The banks are numbered from 0 to one less than the value found in Core::X86::Msr::MCG\_CAP[Count]. For example, if the Count field indicates five banks are supported, they are numbered MC0 through MC4.

#### 3.1.5.5 Determining Bank Type

To determine which type of block is mapped to an MCA bank, software can query the MCA\_IPID register within that bank. This register exists when MCA\_CONFIG[McaX] == 1 in a given bank.

MCA\_IPID[HardwareID] provides the block type for the block that contains this MCA bank. For blocks that contain multiple MCA bank types (e.g., CPU cores), MCA\_IPID[McaType] provides an identifier for the type of MCA bank. MCA\_IPID[McaType] values are specific to a given MCA\_IPID[HardwareID]. Therefore, an MCA bank type can be identified by the value of {MCA\_IPID[Hwid], MCA\_IPID[McaType]}. For instance, the CPU core's LS bank is identified by MCA::LS::MCA\_IPID\_LS[HardwareID] == 176 and MCA::LS::MCA\_IPID\_LS[McaType] == 0. An MCA\_IPID[HardwareID] value of 0 indicates an unpopulated MCA bank that is ensured to be RAZ/WRIG.

MCA\_IPID[InstanceId] provides a unique instance number to allow software to differentiate blocks with multiple identical instances within a processor. MCA\_IPID[InstanceId] values are processor-specific and are not ensured to be stable across different processor generations.

#### 3.1.5.6 Recognizing Error Type

Software can use the combination of MCA\_IPID[Hwid, McaType] and MCA\_STATUS[ErrorCodeExt] to recognize a specific error type.

### 3.1.5.7 Machine Check Error Handling

A machine check handler is invoked to handle an exception for a particular thread. The information needed by the machine check handler is not shared with other threads, so no cross-thread coordination or special handling is required. Specifically, all MCA banks are only visible from a single thread, so software on a single thread can access each bank through MSR space without contention from other threads.

At a minimum, the machine check handler must be capable of logging error information for later examination. The handler should log as much information as is needed to diagnose the error. More thorough exception handler implementations can analyze errors to determine if each error is recoverable by software. If a recoverable error is identified, the exception handler can attempt to correct the error and restart the interrupted program. An error may not be recoverable for the process or virtual machine it directly affects, but may be containable, so that other processes or virtual machines in the system are unaffected and system operation is recovered.

Machine check exception handlers that attempt to recover must be thorough in their analysis and the corrective actions they take. The following guidelines should be used when writing such a handler:

- Data collection:
  - Read Core::X86::Msr::MCG\_CAP[Count] to determine the number of status registers visible to the logical core.
  - All status registers in all error reporting banks must be examined to identify the cause of the machine check exception.
  - Check the valid bit in each status register (MCA\_STATUS[Val]). The remainder of the status register should be examined only when its valid bit is set.
  - When identifying the error condition and determining how to handle the error, portable exception handlers should examine only DOER fields in machine check registers.
  - Error handlers should collect all available MCA information, but should only interrogate details to the level which affects their actions. Lower level details may be useful for diagnosis and root cause analysis, but not for error handling.
  - Error handlers should save the values in MCA\_ADDR, MCA\_MISC0, and MCA\_SYND even if MCA\_STATUS[AddrV], MCA\_STATUS[MiscV], and MCA\_STATUS[SyndV] are zero. Error handlers should save the values in MCA\_MISC[4:1] if the registers exist.
- DOER Error Management:
  - Check MCA\_STATUS[PCC].
    - If PCC is set, error recovery is not possible. The handler should log the error information and terminate the system. If PCC is clear, the handler may continue with the following recovery steps.
  - Check MCA\_STATUS[UC].
    - If UC is set, the processor did not correct the error. Continue with the following recovery steps.
      - If MCA Overflow Recovery is not supported, and MCA\_STATUS[Overflow] == 1, error recovery is not possible; follow the steps for PCC = 1. See 3.1.3.6 [MCA Overflow Recovery].
      - If MCA Recovery is not supported, error recovery is not possible; follow the steps for PCC = 1. See 3.1.3.7 [MCA Recovery].
      - If MCA Recovery is supported:
        - Check MCA\_STATUS[TCC].
          - If TCC is set, the context of the process thread executing on the interrupted logical core may be corrupt and the thread cannot be recovered. The rest of the system is unaffected; it is possible to terminate only the affected process thread.
          - If TCC is clear, the context of the process thread executing on the interrupted logical core is not corrupt. Recovery of the process thread may be possible, but only if the uncorrected error condition is first corrected by software. Otherwise, the interrupted process thread must be terminated.

- Legacy exception handlers can check Core::X86::Msr::MCG\_STAT[RIPV] and Core::X86::Msr::MCG\_STAT[EIPV] in place of MCA\_STATUS[TCC]. If RIPV == EIPV == 1, the interrupted program can be restarted reliably. Otherwise, the program cannot be restarted reliably.
- If UC is clear, the processor either corrected or deferred the error and no software action is needed. The handler can log the error information and continue process execution.
- Exit:
  - When an exception handler is able to successfully log an error condition, clear the MCA\_STATUS registers prior to exiting the machine check handler.
  - Prior to exiting the machine check handler, clear Core::X86::Msr::MCG\_STAT[MCIP]. MCIP indicates that a machine check exception is in progress. If this bit is set when another machine check exception occurs, the processor enters the shutdown state.

## 3.2 Machine Check Architecture Implementation

### 3.2.1 Implemented Machine Check Banks

Table 31: Blocks Capable of Supporting MCA Banks

Acronym	Block Function
LS	Load-Store Unit
IF	Instruction Fetch Unit
L2	L2 Cache Unit
DE	Decode Unit
EX	Execution Unit
FP	Floating-Point Unit
L3	L3 Cache Unit
PIE	Power Management, Interrupts, Etc.
CS	Coherent Station
UMC	Unified Memory Controller
NBIO	Northbridge IO Unit
PCS_GMI	GMI Controller
KPX_SERDES	High Speed Interface Unit
KPX_GMI	High Speed Interface Unit (GMI)

Table 32: Mapping of Blocks to MCA\_IPID[HwId] and MCA\_IPID[McaType]

Block	Hardware ID	MCA Type
LS	0xb0	0x0
IF	0xb0	0x1
L2	0xb0	0x2
L3	0xb0	0x7
MP5	0x1	0x2
PCS_GMI	0x241	0x0
KPX_GMI	0x269	0x0
UMC	0x96	0x1
NBIO	0x18	0x0

PIE	0x2e	0x1
KPX_SERDES	0x259	0x0
CS	0x2e	0x2
EX	0xb0	0x5
FP	0xb0	0x6
DE	0xb0	0x3

### 3.2.2 Implemented Machine Check Bank Registers

Table 33 [Legacy MCA Registers] provides links to the description of each block's Legacy MCA registers. Table 34 [MCAX Registers] provides links to the description of each block's MCA Extension Registers.

*Table 33: Legacy MCA Registers*

Block	MCA Register				
	CTL	STATUS	ADDR	MISC	CTL_MASK
LS	MCA::LS::MCA_CTL_LS	MCA::LS::MCA_STATUS_LS	MCA::LS::MCA_ADDR_LS	MCA::LS::MCA_MISC0_LS	MCA::LS::MCA_CTL_MASK_LS
IF	MCA::IF::MCA_CTL_IF	MCA::IF::MCA_STATUS_IF	MCA::IF::MCA_ADDR_IF	MCA::IF::MCA_MISC0_IF	MCA::IF::MCA_CTL_MASK_IF
L2	MCA::L2::MCA_CTL_L2	MCA::L2::MCA_STATUS_L2	MCA::L2::MCA_ADDR_L2	MCA::L2::MCA_MISC0_L2	MCA::L2::MCA_CTL_MASK_L2
DE	MCA::DE::MCA_CTL_DE	MCA::DE::MCA_STATUS_DE	MCA::DE::MCA_ADDR_DE	MCA::DE::MCA_MISC0_DE	MCA::DE::MCA_CTL_MASK_DE
EX	MCA::EX::MCA_CTL_EX	MCA::EX::MCA_STATUS_EX	MCA::EX::MCA_ADDR_EX	MCA::EX::MCA_MISC0_EX	MCA::EX::MCA_CTL_MASK_EX
FP	MCA::FP::MCA_CTL_FP	MCA::FP::MCA_STATUS_FP	MCA::FP::MCA_ADDR_FP	MCA::FP::MCA_MISC0_FP	MCA::FP::MCA_CTL_MASK_FP
L3	MCA::L3::MCA_CTL_L3	MCA::L3::MCA_STATUS_L3	MCA::L3::MCA_ADDR_L3	MCA::L3::MCA_MISC0_L3	MCA::L3::MCA_CTL_MASK_L3
PIE	MCA::PIE::MCA_CTL_PIE	MCA::PIE::MCA_STATUS_PIE	MCA::PIE::MCA_ADDR_PIE	MCA::PIE::MCA_MISC0_PIE	MCA::PIE::MCA_CTL_MASK_PIE
CS	MCA::CS::MCA_CTL_CS	MCA::CS::MCA_STATUS_CS	MCA::CS::MCA_ADDR_CS	MCA::CS::MCA_MISC0_CS	MCA::CS::MCA_CTL_MASK_CS
UMC	MCA::UMC::MCA_CTL_UMC	MCA::UMC::MCA_STATUS_UMC	MCA::UMC::MCA_ADDR_UMC	MCA::UMC::MCA_MISC0_UMC MCA::UMC::MCA_MISC1_UMC	MCA::UMC::MCA_CTL_MASK_UMC
NBIO	MCA::NBIO::MCA_CTL_NBIO	MCA::NBIO::MCA_STATUS_NBIO	MCA::NBIO::MCA_ADDR_NBIO	MCA::NBIO::MCA_MISC0_NBIO	MCA::NBIO::MCA_CTL_MASK_NBIO
KPX_SERDES	MCA::KPX::SERDES::MCA_CTL_KPX_SERDES	MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES	MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES	MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES	MCA::KPX::SERDES::MCA_CTL_MASK_KPX_SERDES

*Table 34: MCAX Registers*

Block	MCA Register				
	CONFIG	IPID	SYND	DESTAT	DEADDR
LS	MCA::LS::MCA_CONFIG_LS	MCA::LS::MCA_IPID_LS	MCA::LS::MCA_SYND_LS	MCA::LS::MCA_DESTAT_LS	MCA::LS::MCA_DEADDR_LS
IF	MCA::IF::MCA_CONFIG_IF	MCA::IF::MCA_IPID_IF	MCA::IF::MCA_SYND_IF	--	--
L2	MCA::L2::MCA_CONFIG_L2	MCA::L2::MCA_IPID_L2	MCA::L2::MCA_SYND_L2	MCA::L2::MCA_DESTAT_L2	MCA::L2::MCA_DEADDR_L2
DE	MCA::DE::MCA_CONFIG_DE	MCA::DE::MCA_IPID_DE	MCA::DE::MCA_SYND_DE	--	--
EX	MCA::EX::MCA_CONFIG_EX	MCA::EX::MCA_IPID_EX	MCA::EX::MCA_SYND_EX	--	--
FP	MCA::FP::MCA_CONFIG_FP	MCA::FP::MCA_IPID_FP	MCA::FP::MCA_SYND_FP	--	--
L3	MCA::L3::MCA_CONFIG_L3	MCA::L3::MCA_IPID_L3	MCA::L3::MCA_SYND_L3	MCA::L3::MCA_DESTAT_L3	MCA::L3::MCA_DEADDR_L3

PIE	MCA::PIE::MCA_CONFIG_PIE	MCA::PIE::MCA_IPID_PIE	MCA::PIE::MCA_SYND_PIE	MCA::PIE::MCA_DESTAT_PIE	MCA::PIE::MCA_DEADDR_PIE
CS	MCA::CS::MCA_CONFIG_CS	MCA::CS::MCA_IPID_CS	MCA::CS::MCA_SYND_CS	MCA::CS::MCA_DESTAT_CS	MCA::CS::MCA_DEADDR_CS
UMC	MCA::UMC::MCA_CONFIG_UMC	MCA::UMC::MCA_IPID_UMC	MCA::UMC::MCA_SYND_UMC	MCA::UMC::MCA_DESTAT_UMC	MCA::UMC::MCA_DEADDR_UMC
NBIO	MCA::NBIO::MCA_CONFIG_NBIO	MCA::NBIO::MCA_IPID_NBIO	MCA::NBIO::MCA_SYND_NBIO	MCA::NBIO::MCA_DESTAT_NBIO	
KPX_SERDES	MCA::KPX::SERDES::MCA_CONFIG_KPX_SERDES	MCA::KPX::SERDES::MCA_IPID_KPX_SERDES	MCA::KPX::SERDES::MCA_SYND_KPX_SERDES	--	--

### 3.2.3 Mapping of Banks to Blocks

Table 35 [Core MCA Bank to Block Mapping] shows MCA banks that are present in the address space of every logical core.

*Table 35: Core MCA Bank to Block Mapping*

Bank	Block
0	LS
1	IF
2	L2
3	DE
4	RAZ
5	EX
6	FP

Table 36 [Non-core MCA Bank to Block Mapping] shows MCA banks that are present in the address space of specific logical cores.

*Table 36: Non-core MCA Bank to Block Mapping*

Bank	Thread 0	Thread 2
0	LS	LS
1	IF	IF
2	L2	L2
3	DE	DE
4	RAZ	RAZ
5	EX	EX
6	FP	FP
7	L3	L3
8	L3	L3
9	L3	L3
10	L3	L3
11	L3	L3
12	L3	L3
13	L3	L3
14	L3	L3
15	MP5	MP5

16	PCS_GMI	PCS_GMI
17	PCS_GMI	PCS_GMI
18	KPX_GMI	KPX_GMI
19	KPX_GMI	KPX_GMI
20	RAZ	RAZ
21	UMC	RAZ
22	UMC	RAZ
23	CS	RAZ
24	CS	RAZ
25	RAZ	RAZ
26	RAZ	RAZ
27	NBIO	RAZ
28	RAZ	RAZ
29	RAZ	RAZ
30	PIE	RAZ
31	KPX_SERDES	KPX_SERDES
32	RAZ	RAZ
33	RAZ	RAZ
34	RAZ	RAZ
35	RAZ	RAZ
36	RAZ	RAZ
37	RAZ	RAZ
38	RAZ	RAZ
39	RAZ	RAZ
40	RAZ	RAZ
41	RAZ	RAZ
42	RAZ	RAZ
43	RAZ	RAZ
44	RAZ	RAZ
45	RAZ	RAZ
46	RAZ	RAZ
47	RAZ	RAZ
48	RAZ	RAZ
49	RAZ	RAZ
50	RAZ	RAZ
51	RAZ	RAZ
52	RAZ	RAZ
53	RAZ	RAZ
54	RAZ	RAZ
55	RAZ	RAZ
56	RAZ	RAZ
57	RAZ	RAZ
58	RAZ	RAZ
59	RAZ	RAZ
60	RAZ	RAZ
61	RAZ	RAZ
62	RAZ	RAZ
63	RAZ	RAZ



### 3.2.4 Mapping of Banks to Blocks

Table 37: MCA Bank to Block Mapping

Bank	Block
0	LS
1	IF
2	L2
3	DE
4	RAZ
5	EX/SC
6	FP

### 3.2.5 Decoding Error Type

If a valid error is logged in MCA\_STATUS or MCA\_DESTAT of an MCA bank:

1. Read the values of this bank's MCA\_IPID and MCA\_STATUS registers.
2. Use Table 31 [Blocks Capable of Supporting MCA Banks] to look up the block associated with the values of MCA\_IPID[HwId] and MCA\_IPID[McaType].
3. In 3.2.6 [MCA Banks], find the sub-section associated with the block in error.
4. In this sub-section, find the MCA\_STATUS table.
5. In the table, look up the row associated with the MCA\_STATUS[ErrorCodeExt] value.
6. The error type in this row is the logged error. The MCA\_STATUS, MCA\_ADDR and MCA\_SYND tables contain information associated with this error.
7. If there is an error in both MCA\_STATUS and MCA\_DESTAT, the registers contain the same error if MCA\_STATUS[Deferred] is set. If MCA\_STATUS[Deferred] is not set, MCA\_DESTAT contains information for a different error than MCA\_STATUS.

### 3.2.6 MCA Banks

#### 3.2.6.1 LS

MSR0000_0400...MSRC000_2000 [LS Machine Check Control Thread 0] (MCA::LS::MCA_CTL_LS)	
Read-write. Reset: 0000_0000_0000_0000h.	
0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::LS::MCA_CTL_LS register must be enabled by the corresponding enable bit in Core::X86::Msrr::MCG_CTL. Does not affect error detection, correction, or logging.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst0_n[31:0]_aliasMSRLEGACY; MSR0000_0400	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst0_n[31:0]_aliasMSR; MSRC000_2000	
Bits	Description
63:27	Reserved.
26	<b>Hwa.</b> Read-write. Reset: 0. Hardware Asserts (HWAs)
25	<b>SystemReadDataErrorWcb.</b> Read-write. Reset: 0. System Read Data Error detected by write combine buffer. An error in a read of a line from the data fabric. Possible reasons include decode error and target abort.
24	<b>ScbDataNonCacheable.</b> Read-write. Reset: 0. Error on SCB data for non-cacheable DRAM or IO, discovered at data-pull time
23	<b>ScbData1.</b> Read-write. Reset: 0. Error on SCB data, commit pipe 1, discovered at SCB commit time
22	<b>ScbData0.</b> Read-write. Reset: 0. Error on SCB data, commit pipe 0, discovered at SCB commit time
21	<b>ScbStateAddr.</b> Read-write. Reset: 0. Error on SCB cacheline state (way and moesi state) or address field
20	<b>L2DataErr.</b> Read-write. Reset: 0. L2 Fill Data error.
19	<b>DcTagErr7.</b> Read-write. Reset: 0. DC Tag error type 5.
18	<b>DcTagErr3.</b> Read-write. Reset: 0. DC Tag error type 3.
17	<b>PDC.</b> Read-write. Reset: 0. PDC parity error.
16	<b>L2DTLB.</b> Read-write. Reset: 0. Level 2 TLB parity error.
15	<b>DcTagErr4.</b> Read-write. Reset: 0. DC Tag error type 4.
14	<b>DcDataErr3.</b> Read-write. Reset: 0. DC Data error type 3.
13	<b>DcDataErr2.</b> Read-write. Reset: 0. DC Data error type 2.
12	<b>DcDataErr1.</b> Read-write. Reset: 0. DC Data error type 1 and poison consumption. MCA_STATUS[Poison] is set on poison consumption from L2/L3.
11	<b>DcTagErr2.</b> Read-write. Reset: 0. DC Tag error type 2.
10	<b>SystemReadDataErrorMab.</b> Read-write. Reset: 0. System Read Data Error detected by mab. An error in a read of a line from the data fabric. Possible reasons include decode error and target abort.
9	<b>SystemReadDataErrorUcode.</b> Read-write. Reset: 0. System Read Data Error logged by ucode. An error in a read of a line from the data fabric. Possible reasons include decode error and target abort.
8	<b>IntErrTyp2.</b> Read-write. Reset: 0. Internal error type 2.
7	<b>IntErrTyp1.</b> Read-write. Reset: 0. Internal error type 1.
6	<b>DcTagErr1.</b> Read-write. Reset: 0. DC Tag error type 1.
5	<b>DcTagErr6.</b> Read-write. Reset: 0. DC Tag error type 6.
4	<b>DcTagErr5.</b> Read-write. Reset: 0. DC Tag error type 5.
3	<b>L1DTLB.</b> Read-write. Reset: 0. Level 1 TLB parity error.
2	<b>MAB.</b> Read-write. Reset: 0. Miss address buffer payload parity error.
1	<b>STQ.</b> Read-write. Reset: 0. Store queue parity error.
0	<b>LDQ.</b> Read-write. Reset: 0. Load queue parity error.

**MSR0000\_0001...MSRC000\_2001 [LS Machine Check Status Thread 0] (MCA::LS::MCA\_STATUS\_LS)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSRSLLEGACY; MSR0000\_0001

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0401

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_2001

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::LS::MCA_CTL_LS. This bit is a copy of bit in MCA::LS::MCA_CTL_LS for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::LS::MCA_MISC0_LS. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::LS::MCA_ADDR_LS contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::LS::MCA_STATUS_LS[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::LS::MCA_SYND_LS. If MCA::LS::MCA_SYND_LS[ErrorPriority] is the same as the priority of the error in MCA::LS::MCA_STATUS_LS, then the information in MCA::LS::MCA_SYND_LS is associated with the error in MCA::LS::MCA_STATUS_LS. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor.

	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::LS::MCA_ADDR_LS[ErrorAddr]. A value of 0 indicates that MCA::LS::MCA_ADDR_LS[63:0] contains a valid byte address. A value of 6 indicates that MCA::LS::MCA_ADDR_LS[63:6] contains a valid cache line address and that MCA::LS::MCA_ADDR_LS[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::LS::MCA_ADDR_LS[63:12] contain a valid 4KB memory page and that MCA::LS::MCA_ADDR_LS[11:0] should be ignored by error handling software.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::LS::MCA_CTL_LS enables error reporting for the logged error.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 38: MCA\_STATUS\_LS

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
LDQ	0x0	1	1	1	0	0	0
STQ	0x1	1	1	1	0	0	0
MAB	0x2	1	1	1	0	0	0
L1DTLB	0x3	1	1	1	0	0	1
DcTagErr5	0x4	1	1	1	0	0	0
DcTagErr6	0x5	1	1	1	0	0	0
DcTagErr1	0x6	1	1	1	0	0	0
IntErrTyp1	0x7	1	1	1	0	0	0
IntErrTyp2	0x8	0/1	0/1	0/1	0	0	0
SystemRead DataErrorUc	0x9	1	1	1	0	0	0/1

ode							
SystemRead DataErrorMa b	0xa	1	1	1	0	0	0/1
DcTagErr2	0xb	0	0	0	0	0	0
DcDataErr1	0xc	0/1	0	0/1	0	0/1	1
DcDataErr2	0xd	0	0	0	0/1	0	1
DcDataErr3	0xe	0	0	0	0/1	0	0/1
DcTagErr4	0xf	0	0	0	1	0	0
L2DTLB	0x10	0	0	0	0	0	0/1
PDC	0x11	0	0	0	0	0	0/1
DcTagErr3	0x12	0	0	0	0	0	0
DcTagErr7	0x13	0	0	0	0	0	0
L2DataErr	0x14	0	0	0	0	0	0
ScbStateAdd r	0x15	1	1	1	0	0	0
ScbData0	0x16	0	0	0	1	1	0
ScbData1	0x17	0	0	0	1	1	0
ScbDataNon Cacheable	0x18	0	0	0	1	1	0
SystemRead DataErrorW cb	0x19	1	1	1	0	0	0/1
Hwa	0x1a	1	1	1	0	0	0
Reserved	0x1b	0	0	0	0	0	0

**MSR0000\_0000...MSRC000\_2002 [LS Machine Check Address Thread 0] (MCA::LS::MCA\_ADDR\_LS)**

Read-write. Reset: Cold,0000\_0000\_0000\_0000h.

MCA::LS::MCA\_ADDR\_LS stores an address and other information associated with the error in MCA::LS::MCA\_STATUS\_LS. The register is only meaningful if MCA::LS::MCA\_STATUS\_LS[Val]=1 and MCA::LS::MCA\_STATUS\_LS[AddrV]=1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSRLSLEGACY; MSR0000\_0000

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0402

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_2002

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write. Reset: Cold,0000_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::LS::MCA_STATUS_LS. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 39: MCA\_ADDR\_LS

Error Type	Bits	Description
LDQ	[63:0]	Reserved
STQ	[63:0]	Reserved
MAB	[63:0]	Reserved
L1DTLB	[63:48] [47:12] [11:0]	Reserved Virtual Address Reserved
DcTagErr5	[63:0]	Reserved
DcTagErr6	[63:0]	Reserved
DcTagErr1	[63:0]	Reserved

IntErrTyp1	[63:0]	Reserved
IntErrTyp2	[63:0]	Reserved
SystemReadDataErrorUcode	[63:48] [47:6]	Reserved Physical Address
SystemReadDataErrorMab	[63:0]	Reserved
DcTagErr2	[63:0]	Reserved
DcDataErr1	[63:48] [47:6] [5:1]	Reserved Physical Address MCA_STATUS_LS[Poison]=1 ? 5'b0 : Physical Address
DcDataErr2	[63:48] [47:1]	Reserved Physical Address
DcDataErr3	[63:48] [47:1]	Reserved Physical Address
DcTagErr4	[63:0]	Reserved
L2DTLB	[63:48] [47:12] [11:0]	Reserved Virtual Address Reserved
PDC	[63:48] [47:12] [11:0]	Reserved Virtual Address Reserved
DcTagErr3	[63:0]	Reserved
DcTagErr7	[63:0]	Reserved
L2DataErr	[63:0]	Reserved
ScbStateAddr	[63:0]	Reserved
ScbData0	[63:0]	Reserved
ScbData1	[63:0]	Reserved
ScbDataNonCacheable	[63:0]	Reserved
SystemReadDataErrorWcb	[63:0]	Reserved
Hwa	[63:0]	Reserved
Reserved	[63:0]	Reserved

**MSR0000\_0403...MSRC000\_2003 [LS Machine Check Miscellaneous 0 Thread 0] (MCA::LS::MCA\_MISC0\_LS)**

Log miscellaneous information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0403

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_2003

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::LS::MCA_MISC0_LS[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::LS::MCA_MISC0_LS[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::LS::MCA_MISC0_LS[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::LS::MCA_MISC0_LS[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::LS::MCA_MISC0_LS[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::LS::MCA_MISC0_LS[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.



**MSRC000\_2004 [LS Machine Check Configuration Thread 0] (MCA::LS::MCA\_CONFIG\_LS)**

Reset: 0000\_0000\_0000\_0125h.

Controls configuration of the associated machine check bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_2004

Bits	Description
63:41	Reserved.
40	<b>IntEn.</b> Read-write. Reset: 0. Init: BIOS,0. 1=When set, this bank will generate corrected error interrupts.
39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:35	Reserved.
34	<b>LogDeferredInMcaStat.</b> Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::LS::MCA_STATUS_LS and MCA::LS::MCA_ADDR_LS in addition to MCA::LS::MCA_DESTAT_LS and MCA::LS::MCA_DEADDR_LS. 0=Only log deferred errors in MCA::LS::MCA_DESTAT_LS and MCA::LS::MCA_DEADDR_LS. This bit does not affect logging of deferred errors in MCA::LS::MCA_SYND_LS, MCA::LS::MCA_MISC0_LS.
33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:11	Reserved.
10	<b>IntPresent.</b> Read-only. Reset: 0. 1=This bank can be configured to trigger a corrected error interrupt using MCA::LS::MCA_CONFIG_LS[IntEn].
9	<b>McaFruTextInMca.</b> Read-write. Reset: 0. Init: BIOS,0. 1=FruText is reported McaSynd1/McaSynd2 registers
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::LS::MCA_CONFIG_LS[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::LS::MCA_CONFIG_LS[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::LS::MCA_CONFIG_LS[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::LS::MCA_CONFIG_LS[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::LS::MCA_DESTAT_LS and MCA::LS::MCA_DEADDR_LS are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::LS::MCA_MISC0_LS[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::LS::MCA_STATUS_LS[TCC] is present.



**MSRC000\_2005 [LS IP Identification Thread 0] (MCA::LS::MCA\_IPID\_LS)**

Reset: 0000\_00B0\_0000\_0000h.

The MCA::LS::MCA\_IPID\_LS register is used by software to determine what IP type and revision is associated with the MCA bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_2005

Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_2006 [LS Machine Check Syndrome Thread 0] (MCA::LS::MCA\_SYND\_LS)**

Read-write. Reset: Cold,0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::LS::MCA\_STATUS\_LS Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_2006

Bits	Description
63:32	<b>Syndrom.</b> Read-write. Reset: Cold,0000_0000h. Contains the syndrome, if any, associated with the error logged in MCA::LS::MCA_STATUS_LS. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::LS::MCA_SYND_LS[Length]. The Syndrome field is only valid when MCA::LS::MCA_SYND_LS[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority.</b> Read-write. Reset: Cold,0h. Encodes the priority of the error logged in MCA::LS::MCA_SYND_LS. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length.</b> Read-write. Reset: Cold,00h. Specifies the length in bits of the syndrome contained in MCA::LS::MCA_SYND_LS[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::LS::MCA_SYND_LS. For example, a syndrome length of 9 means that MCA::LS::MCA_SYND_LS[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation.</b> Read-write. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 40 [MCA_SYND_LS].

Table 40: MCA\_SYND\_LS

Error Type	Bits	Description
LDQ	[17:0]	Reserved
STQ	[17:0]	Reserved
MAB	[17:0]	Reserved
L1DTLB	[17:0]	Reserved
DcTagErr5	[17:16] [15:8] [7:0]	2'b11 Index Way
DcTagErr6	[17:16] [15:8] [7:0]	2'b11 Index Way
DcTagErr1	[17:16] [15:8] [7:0]	2'b11 Index Way
IntErrTyp1	[17:11] [10]	Reserved Thread ID

	[9:0]	Reserved
IntErrTyp2	[17:12] [11] [10:1] [0]	Reserved Thread ID Reserved Reserved
SystemReadDataErrorUcode	[17:2] [1:0]	Reserved 2'b00 = Master Abort; 2'b01 = Target Abort; 2'b10 = Transaction Error; 2'b11 = Protection Violation
SystemReadDataErrorMab	[17:12] [11:8] [7:2] [1:0]	Reserved DC way holding the miss address where the error occurred Address [11:6] of error 2'b00 = Master Abort; 2'b01 = Target Abort; 2'b10 = Transaction Error; 2'b11 = Protection Violation
DcTagErr2	[17:16] [15:8] [7:0]	2'b11 Index Way
DcDataErr1	[17:16] [15:8] [7:0]	MCA_STATUS_LS[Poison]=1 ? 2'b00 : 2'b11 Index Way
DcDataErr2	[17:16] [15:8] [7:0]	2'b11 Index Way
DcDataErr3	[17:16] [15:14] [13:8] [7:3] [2:0]	2'b11 Reserved Index Physical Address[5:1] Way
DcTagErr4	[17:16] [15:8] [7:0]	Reserved Index Way
L2DTLB	[17:16] [15] [14:8] [7:4] [3:0]	2'b11 Reserved Reserved Reserved Reserved
PDC	[17:0]	Reserved
DcTagErr3	[17:16] [15:8] [7:0]	2'b11 Index Way
DcTagErr7	[17:16] [15:8] [7:0]	2'b11 Index Way
L2DataErr	[17:0]	Reserved
ScbStateAddr	[17:0]	Reserved
ScbData0	[17:0]	Reserved
ScbData1	[17:0]	Reserved
ScbDataNonCacheable	[17:0]	Reserved
SystemReadDataErrorWcb	[17:2] [1:0]	Reserved 2'b00 = Master Abort; 2'b01 = Target Abort; 2'b10 = Transaction Error; 2'b11 = Protection Violation

Hwa	[17:0]	Reserved
Reserved	[17:0]	Reserved

**MSRC000\_2008 [LS Machine Check Deferred Error Status Thread 0] (MCA::LS::MCA\_DESTAT\_LS)**

Read-write. Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_2008

Bits	Description
63	<b>Val.</b> Read-write. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
58	<b>AddrV.</b> Read-write. Reset: Cold,0. 1=MCA::LS::MCA_DEADDR_LS contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
53	<b>SyndV.</b> Read-write. Reset: Cold,0. 1=This error logged information in MCA::LS::MCA_SYND_LS. If MCA::LS::MCA_SYND_LS[ErrorPriority] is the same as the priority of the error in MCA::LS::MCA_STATUS_LS, then the information in MCA::LS::MCA_SYND_LS is associated with the error in MCA::LS::MCA_DESTAT_LS.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h. MCA_DEFSTAT Register Reserved bits.
44	<b>Deferred.</b> Read-write. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h. MCA_DEFSTAT Register Reserved bits.
29:24	<b>AddrLsb.</b> Read-write. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::LS::MCA_ADDR_LS[ErrorAddr]. A value of 0 indicates that MCA::LS::MCA_ADDR_LS[63:0] contains a valid byte address. A value of 6 indicates that MCA::LS::MCA_ADDR_LS[63:6] contains a valid cache line address and that MCA::LS::MCA_ADDR_LS[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::LS::MCA_ADDR_LS[63:12] contain a valid 4KB memory page and that MCA::LS::MCA_ADDR_LS[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
21:16	<b>ErrorCodeExt.</b> Read-write. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write. Reset: Cold,0000h. Error code for this error.

**MSRC000\_2009 [LS Deferred Error Address Thread 0] (MCA::LS::MCA\_DEADDR\_LS)**

Read-write. Reset: Cold,0000\_0000\_0000\_0000h.

The MCA::LS::MCA\_DEADDR\_LS register stores the address associated with the error in MCA::LS::MCA\_DESTAT\_LS. The register is only meaningful if MCA::LS::MCA\_DESTAT\_LS[Val]=1 and MCA::LS::MCA\_DESTAT\_LS[AddrV]=1. The lowest valid bit of the address is defined by MCA::LS::MCA\_DESTAT\_LS[AddrLsb].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_2009

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::LS::MCA_DESTAT_LS. The lowest-order valid bit of the address is specified in MCA::LS::MCA_DESTAT_LS[AddrLsb].

**MSRC001\_0400 [LS Machine Check Control Mask Thread 0] (MCA::LS::MCA\_CTL\_MASK\_LS)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC001\_0400

Bits	Description
63:27	Reserved.
26	<b>Hwa.</b> Read-write. Reset: 0. Hardware Asserts (HWAs)
25	<b>SystemReadDataErrorWcb.</b> Read-write. Reset: 0. Init: BIOS,1. System Read Data Error detected by write combine buffer. An error in a read of a line from the data fabric. Possible reasons include decode error and target abort.
24	<b>ScbDataNonCacheable.</b> Read-write. Reset: 0. Error on SCB data for non-cacheable DRAM or IO, discovered at data-pull time
23	<b>ScbData1.</b> Read-write. Reset: 0. Error on SCB data, commit pipe 1, discovered at SCB commit time
22	<b>ScbData0.</b> Read-write. Reset: 0. Error on SCB data, commit pipe 0, discovered at SCB commit time
21	<b>ScbStateAddr.</b> Read-write. Reset: 0. Error on SCB cacheline state (way and moesi state) or address field
20	<b>L2DataErr.</b> Read-write. Reset: 0. L2 Fill Data error.
19	<b>DcTagErr7.</b> Read-write. Reset: 0. DC Tag error type 5.
18	<b>DcTagErr3.</b> Read-write. Reset: 0. DC Tag error type 3.
17	<b>PDC.</b> Read-write. Reset: 0. PDC parity error. MCA_ADDR_LS logs a virtual address.
16	<b>L2DTLB.</b> Read-write. Reset: 0. Level 2 TLB parity error. MCA_ADDR_LS logs a virtual address.
15	<b>DcTagErr4.</b> Read-write. Reset: 0. DC Tag error type 4.
14	<b>DcDataErr3.</b> Read-write. Reset: 0. DC Data error type 3.
13	<b>DcDataErr2.</b> Read-write. Reset: 0. DC Data error type 2.
12	<b>DcDataErr1.</b> Read-write. Reset: 0. DC Data error type 1 and poison consumption. MCA_STATUS[Poison] is set on poison consumption from L2/L3.
11	<b>DcTagErr2.</b> Read-write. Reset: 0. DC Tag error type 2.
10	<b>SystemReadDataErrorMab.</b> Read-write. Reset: 0. Init: BIOS,1. System Read Data Error detected by mab. An error in a read of a line from the data fabric. Possible reasons include decode error and target abort.
9	<b>SystemReadDataErrorUcode.</b> Read-write. Reset: 0. Init: BIOS,1. System Read Data Error logged by ucode. An error in a read of a line from the data fabric. Possible reasons include decode error and target abort.
8	<b>IntErrTyp2.</b> Read-write. Reset: 0. Internal error type 2.
7	<b>IntErrTyp1.</b> Read-write. Reset: 0. Internal error type 1.
6	<b>DcTagErr1.</b> Read-write. Reset: 0. DC Tag error type 1.
5	<b>DcTagErr6.</b> Read-write. Reset: 0. DC Tag error type 6.
4	<b>DcTagErr5.</b> Read-write. Reset: 0. DC Tag error type 5.
3	<b>L1DTLB.</b> Read-write. Reset: 0. Level 1 TLB parity error.
2	<b>MAB.</b> Read-write. Reset: 0. Miss address buffer payload parity error.
1	<b>STQ.</b> Read-write. Reset: 0. Store queue parity error.
0	<b>LDQ.</b> Read-write. Reset: 0. Load queue parity error.

**MSRC000\_200E [LS Machine Check Syndrome Extended Thread 0] (MCA::LS::MCA\_SYND1\_LS)**

Read-write. Reset: Cold,0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::LS::MCA\_STATUS\_LS Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_200E

Bits	Description
63:0	<b>Syndrome.</b> Read-write. Reset: Cold,0000_0000_0000_0000h. The MCA::LS::MCA_SYND1_LS register stores information associated with the error in MCA::LS::MCA_STATUS_LS or MCA_DESTAT. The register is meaningful if MCA::LS::MCA_STATUS_LS[SyndV]=1. When MCA::LS::MCA_CONFIG_LS[McaFruTextInMca]=1, MCA::LS::MCA_SYND1_LS stores ASCII FruText associated with the error.

**MSRC000\_200F [LS Machine Check Syndrome Extended Thread 0] (MCA::LS::MCA\_SYND2\_LS)**

Read-write. Reset: Cold,0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::LS::MCA\_STATUS\_LS Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst0\_n[31:0]\_aliasMSR; MSRC000\_200F

Bits	Description
63:0	<b>Syndrome.</b> Read-write. Reset: Cold,0000_0000_0000_0000h. The MCA::LS::MCA_SYND2_LS register stores information associated with the error in MCA::LS::MCA_STATUS_LS or MCA_DESTAT. The register is meaningful if MCA::LS::MCA_STATUS_LS[SyndV]=1. When MCA::LS::MCA_CONFIG_LS[McaFruTextInMca]=1, MCA::LS::MCA_SYND2_LS stores ASCII FruText associated with the error.

**3.2.6.2 IF****MSR0000\_0404...MSRC000\_2010 [IF Machine Check Control Thread 0] (MCA::IF::MCA\_CTL\_IF)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::IF::MCA\_CTL\_IF register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0404

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC000\_2010

Bits	Description
63:20	Reserved.
19	<b>RSVD19.</b> Read-write. Reset: 0.
18	<b>CtMceError.</b> Read-write. Reset: 0. CT MCE
17	<b>RSVD17.</b> Read-write. Reset: 0. Reserved. Will never trigger.
16	<b>L2TlbMultiHit.</b> Read-write. Reset: 0. L2-TLB Multi-Hit
15	<b>L1TlbMultiHit.</b> Read-write. Reset: 0. L1-TLB Multi-Hit.
14	<b>HwAssert.</b> Read-write. Reset: 0. Hardware Assertion Error.
13	<b>SystemReadDataError.</b> Read-write. Reset: 0. System Read Data Error. An error in a demand fetch of a line. Possible reasons include decode error and target abort.
12	<b>L2RespPoison.</b> Read-write. Reset: 0. L2 Cache Response Poison Error. Error is the result of consuming poison data.
11	<b>L2BtbMultiHit.</b> Read-write. Reset: 0. L2 BTB Multi-Match Error.
10	<b>L1BtbMultiHit.</b> Read-write. Reset: 0. L1 BTB Multi-Match Error.
9	<b>BpqSnpParT1.</b> Read-write. Reset: 0. BPQ Thread 1 Snoop Parity Error.
8	<b>BpqSnpParT0.</b> Read-write. Reset: 0. BPQ Thread 0 Snoop Parity Error.
7	<b>L2ItlbParity.</b> Read-write. Reset: 0. L2 ITLB Parity Error.
6	<b>L1ItlbParity.</b> Read-write. Reset: 0. L1 ITLB Parity Error.
5	<b>RSVD5.</b> Read-write. Reset: 0. Reserved. Will never trigger.
4	<b>DqParity.</b> Read-write. Reset: 0. Decoupling Queue PhysAddr Parity Error.
3	<b>DataParity.</b> Read-write. Reset: 0. IC Data Array Parity Error.
2	<b>TagParity.</b> Read-write. Reset: 0. IC Full Tag Parity Error.
1	<b>TagMultiHit.</b> Read-write. Reset: 0. IC Microtag or Full Tag Multi-hit Error.
0	<b>OcUtagParity.</b> Read-write. Reset: 0. Op Cache Microtag Probe Port Parity Error.

**MSR0000\_0405...MSRC000\_2011 [IF Machine Check Status Thread 0] (MCA::IF::MCA\_STATUS\_IF)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0405

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC000\_2011

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::IF::MCA_CTL_IF. This bit is a copy of bit in MCA::IF::MCA_CTL_IF for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::IF::MCA_MISC0_IF. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::IF::MCA_ADDR_IF contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::IF::MCA_STATUS_IF[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::IF::MCA_SYND_IF. If MCA::IF::MCA_SYND_IF[ErrorPriority] is the same as the priority of the error in MCA::IF::MCA_STATUS_IF, then the information in MCA::IF::MCA_SYND_IF is associated with the error in MCA::IF::MCA_STATUS_IF. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.



44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::IF::MCA_ADDR_IF[ErrorAddr]. A value of 0 indicates that MCA::IF::MCA_ADDR_IF[63:0] contains a valid byte address. A value of 6 indicates that MCA::IF::MCA_ADDR_IF[63:6] contains a valid cache line address and that MCA::IF::MCA_ADDR_IF[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::IF::MCA_ADDR_IF[63:12] contain a valid 4KB memory page and that MCA::IF::MCA_ADDR_IF[11:0] should be ignored by error handling software. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::IF::MCA_CTL_IF enables error reporting for the logged error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 41: MCA\_STATUS\_IF

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
OcUtagParity	0x0	1	1	1	0	0	0
TagMultiHit	0x1	0	0	0	0	0	1
TagParity	0x2	0	0	0	0	0	1
DataParity	0x3	0	0	0	0	0	1
DqParity	0x4	1	1	1	0	0	1
RSVD5	0x5	1	1	1	0	0	1
L1ItlbParity	0x6	1	1	1	0	0	1
L2ItlbParity	0x7	0	0	0	0	0	1
BpqSnpParT0	0x8	0	0	0	0	0	0
BpqSnpParT	0x9	0	0	0	0	0	0

1							
L1BtbMulti Hit	0xa	0	0	0	0	0	0
L2BtbMulti Hit	0xb	0	0	0	0	0	0
L2RespPoison	0xc	1	0	1	0	1	1
SystemReadDataError	0xd	1	0	1	0	0	1
HwAssert	0xe	1	1	1	0	0	0
L1TlbMulti Hit	0xf	1	1	1	0	0	1
L2TlbMulti Hit	0x10	0	0	0	0	0	1
RSVD17	0x11	1	1	1	0	0	0
CtMceError	0x12	1	1	1	0	0	0
RSVD19	0x13	0	0	0	0	0	1

**MSR0000\_0406...MSRC000\_2012 [IF Machine Check Address Thread 0] (MCA::IF::MCA\_ADDR\_IF)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

MCA::IF::MCA\_ADDR\_IF stores an address and other information associated with the error in MCA::IF::MCA\_STATUS\_IF. The register is only meaningful if MCA::IF::MCA\_STATUS\_IF[Val]=1 and MCA::IF::MCA\_STATUS\_IF[AddrV]=1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0406

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC000\_2012

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::IF::MCA_STATUS_IF. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 42: MCA\_ADDR\_IF

Error Type	Bits	Description
OcUtagParity	[63:0]	Reserved
TagMultiHit	[63:52] [51:0]	Reserved Physical Address
TagParity	[63:52] [51:0]	Reserved Physical Address
DataParity	[63:52] [51:0]	Reserved Physical Address
DqParity	[63:52] [51:0]	Reserved Physical Address
RSVD5	[63:57] [56:12] [11:0]	Reserved Linear Address Reserved
L1ItlbParity	[63:57] [56:12] [11:0]	Reserved Linear Address Reserved
L2ItlbParity	[63:57] [56:12] [11:0]	Reserved Linear Address Reserved



BpqSnpParT0	[63:0]	Reserved
BpqSnpParT1	[63:0]	Reserved
L1BtbMultiHit	[63:0]	Reserved
L2BtbMultiHit	[63:0]	Reserved
L2RespPoison	[63:52] [51:5] [4:0]	Reserved Physical Address Reserved
SystemReadDataError	[63:52] [51:5] [4:0]	Reserved Physical Address Reserved
HwAssert	[56:0]	Reserved
L1TlbMultiHit	[56:0]	VA
L2TlbMultiHit	[56:0]	VA
RSVD17	[56:0]	Reserved
CtMceError	[56:0]	Reserved
RSVD19	[63:0]	Reserved

**MSR0000\_0407...MSRC000\_2013 [IF Machine Check Miscellaneous 0 Thread 0] (MCA::IF::MCA\_MISC0\_IF)**

Log miscellaneous information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0407

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC000\_2013

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::IF::MCA_MISC0_IF[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::IF::MCA_MISC0_IF[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::IF::MCA_MISC0_IF[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrlw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::IF::MCA_MISC0_IF[Locked]) ? Read-write : Read-only.
48	<b>Ovrlw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::IF::MCA_MISC0_IF[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::IF::MCA_MISC0_IF[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_2014 [IF Machine Check Configuration Thread 0] (MCA::IF::MCA\_CONFIG\_IF)**

Reset: 0000\_0000\_0000\_0121h.

Controls configuration of the associated machine check bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC000\_2014

Bits	Description
63:41	Reserved.
40	<b>IntEn.</b> Read-write. Reset: 0. Init: BIOS,0. 1=When set, this bank will generate corrected error interrupts.
39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:11	Reserved.
10	<b>IntPresent.</b> Read-only, Volatile. Reset: 0. 1=This bank can be configured to trigger a corrected error interrupt using MCA::IF::MCA_CONFIG_IF[IntEn].
9	<b>McaFruTextInMca.</b> Read-write. Reset: 0. Init: BIOS,0. 1=FruText is reported McaSynd1/McaSynd2 registers
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::IF::MCA_CONFIG_IF[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::IF::MCA_CONFIG_IF[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::IF::MCA_CONFIG_IF[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 0. 1=Deferred errors are supported in this MCA bank, and the LogDeferredInMcaStat field in this register controls the logging behavior of these errors. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::IF::MCA_MISC0_IF[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::IF::MCA_STATUS_IF[TCC] is present.

**MSRC000\_2015 [IF IP Identification Thread 0] (MCA::IF::MCA\_IPID\_IF)**

Reset: 0001\_00B0\_0000\_0000h.

The MCA::IF::MCA\_IPID\_IF register is used by software to determine what IP type and revision is associated with the MCA bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC000\_2015

Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0001h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_2016 [IF Machine Check Syndrome Thread 0] (MCA::IF::MCA\_SYND\_IF)**

Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h.	
Logs physical location information associated with error in MCA::IF::MCA_STATUS_IF Thread 0	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst1_n[31:0]_aliasMSR; MSRC000_2016	
Bits	Description
63:32	<b>Syndrone.</b> Read-write, Volatile. Reset: Cold, 0000_0000h. Contains the syndrome, if any, associated with the error logged in MCA::IF::MCA_STATUS_IF. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::IF::MCA_SYND_IF[Length]. The Syndrome field is only valid when MCA::IF::MCA_SYND_IF[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority.</b> Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::IF::MCA_SYND_IF. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length.</b> Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::IF::MCA_SYND_IF[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::IF::MCA_SYND_IF. For example, a syndrome length of 9 means that MCA::IF::MCA_SYND_IF[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation.</b> Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 43 [MCA_SYND_IF].

Table 43: MCA\_SYND\_IF

Error Type	Bits	Description
OcUtagParity	[17:6] [5:0]	Reserved Index
TagMultiHit	[17:16] [15:8] [8:0]	Reserved Subcache Reserved
TagParity	[17:8] [7:0]	Reserved Way
DataParity	[17:16] [15:8] [8:0]	Reserved Subcache Way
DqParity	[17:0]	Reserved
RSVD5	[17:4] [3:0]	Reserved Reserved
L1ItlbParity	[17:6] [5:0]	Reserved Reserved
L2ItlbParity	[17:8] [7:0]	Reserved Reserved
BpqSnpParT0	[17:0]	Reserved
BpqSnpParT1	[17:6] [5:0]	Reserved Index
L1BtbMultiHit	[17:0]	Reserved
L2BtbMultiHit	[17:0]	Reserved
L2RespPoison	[17:0]	Reserved
SystemReadDataError	[17:2] [1:0]	Reserved 2'b00 = Master Abort; 2'b01 = Target Abort; 2'b10 = Transaction Error; 2'b11 = Protection Violation
HwAssert	[17:0]	HwaMcaCode

L1TlbMultiHit	[17:6] [5:0]	Reserved Index
L2TlbMultiHit	[17:8] [7:0]	Reserved Index
RSVD17	[17:0]	Reserved
CtMceError	[17:2] [1:0]	Reserved Thread bit vector
RSVD19	[17:0]	Reserved

**MSRC001\_0401 [IF Machine Check Control Mask Thread 0] (MCA::IF::MCA\_CTL\_MASK\_IF)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC001\_0401

Bits	Description
63:20	Reserved.
19	<b>RSVD19</b> . Read-write. Reset: 0.
18	<b>CtMceError</b> . Read-write. Reset: 0. CT MCE
17	<b>RSVD17</b> . Read-write. Reset: 0. Reserved. Will never trigger.
16	<b>L2TlbMultiHit</b> . Read-write. Reset: 0. L2-TLB Multi-Hit
15	<b>L1TlbMultiHit</b> . Read-write. Reset: 0. L1-TLB Multi-Hit.
14	<b>HwAssert</b> . Read-write. Reset: 0. Hardware Assertion Error.
13	<b>SystemReadDataError</b> . Read-write. Reset: 0. System Read Data Error. An error in a demand fetch of a line. Possible reasons include decode error and target abort.
12	<b>L2RespPoison</b> . Read-write. Reset: 0. L2 Cache Response Poison Error. Error is the result of consuming poison data.
11	<b>L2BtbMultiHit</b> . Read-write. Reset: 0. L2 BTB Multi-Match Error.
10	<b>L1BtbMultiHit</b> . Read-write. Reset: 0. L1 BTB Multi-Match Error.
9	<b>BpqSnpParT1</b> . Read-write. Reset: 0. BPQ Thread 1 Snoop Parity Error.
8	<b>BpqSnpParT0</b> . Read-write. Reset: 0. BPQ Thread 0 Snoop Parity Error.
7	<b>L2ItlbParity</b> . Read-write. Reset: 0. L2 ITLB Parity Error.
6	<b>L1ItlbParity</b> . Read-write. Reset: 0. L1 ITLB Parity Error.
5	<b>RSVD5</b> . Read-write. Reset: 0. Reserved. Will never trigger.
4	<b>DqParity</b> . Read-write. Reset: 0. Decoupling Queue PhysAddr Parity Error.
3	<b>DataParity</b> . Read-write. Reset: 0. IC Data Array Parity Error.
2	<b>TagParity</b> . Read-write. Reset: 0. IC Full Tag Parity Error.
1	<b>TagMultiHit</b> . Read-write. Reset: 0. IC Microtag or Full Tag Multi-hit Error.
0	<b>OcUtagParity</b> . Read-write. Reset: 0. Op Cache Microtag Probe Port Parity Error.

**MSRC000\_201E [IF Machine Check Syndrome Extended Thread 0] (MCA::IF::MCA\_SYND1\_IF)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::IF::MCA\_STATUS\_IF Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC000\_201E

Bits	Description
63:0	<b>Syndrome</b> . Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::IF::MCA_SYND1_IF register stores information associated with the error in MCA::IF::MCA_STATUS_IF or MCA_DESTAT. The register is meaningful if MCA::IF::MCA_STATUS_IF[SyndV]=1. When MCA::IF::MCA_CONFIG_IF[McaFruTextInMca]=1, MCA::IF::MCA_SYND1_IF stores ASCII FruText associated with the error.

**MSRC000\_201F [IF Machine Check Syndrome Extended Thread 0] (MCA::IF::MCA\_SYND2\_IF)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::IF::MCA\_STATUS\_IF Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst1\_n[31:0]\_aliasMSR; MSRC000\_201F

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::IF::MCA_SYND2_IF register stores information associated with the error in MCA::IF::MCA_STATUS_IF or MCA_DESTAT. The register is meaningful if MCA::IF::MCA_STATUS_IF[SyndV]=1. When MCA::IF::MCA_CONFIG_IF[McaFruTextInMca]=1, MCA::IF::MCA_SYND2_IF stores ASCII FruText associated with the error.

**3.2.6.3 L2****MSR0000\_0408...MSRC000\_2020 [L2 Machine Check Control Thread 0] (MCA::L2::MCA\_CTL\_L2)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::L2::MCA\_CTL\_L2 register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0408

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_2020

Bits	Description
63:7	Reserved.
6	<b>Wdt.</b> Read-write. Reset: 0. Reserved
5	<b>StateMachine.</b> Read-write. Reset: 0. Error initiated by programmable state machine.
4	<b>Sdp.</b> Read-write. Reset: 0. SDP Read Response Parity Error
3	<b>Hwa.</b> Read-write. Reset: 0. Hardware Assert Error.
2	<b>Data.</b> Read-write. Reset: 0. L2M Data Array ECC Error.
1	<b>Tag.</b> Read-write. Reset: 0. L2M Tag or State Array ECC Error.
0	<b>MultiHit.</b> Read-write. Reset: 0. L2M Tag Multiple-Way-Hit error.

**MSR0000\_0409...MSRC000\_2021 [L2 Machine Check Status Thread 0] (MCA::L2::MCA\_STATUS\_L2)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0409

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_2021

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::L2::MCA_CTL_L2. This bit is a copy of bit in MCA::L2::MCA_CTL_L2 for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::L2::MCA_MISC0_L2. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::L2::MCA_ADDR_L2 contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::L2::MCA_STATUS_L2[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::L2::MCA_SYND_L2. If MCA::L2::MCA_SYND_L2[ErrorPriority] is the same as the priority of the error in MCA::L2::MCA_STATUS_L2, then the information in MCA::L2::MCA_SYND_L2 is associated with the error in MCA::L2::MCA_STATUS_L2. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.



44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::L2::MCA_ADDR_L2[ErrorAddr]. A value of 0 indicates that MCA::L2::MCA_ADDR_L2[63:0] contains a valid byte address. A value of 6 indicates that MCA::L2::MCA_ADDR_L2[63:6] contains a valid cache line address and that MCA::L2::MCA_ADDR_L2[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::L2::MCA_ADDR_L2[63:12] contain a valid 4KB memory page and that MCA::L2::MCA_ADDR_L2[11:0] should be ignored by error handling software. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::L2::MCA_CTL_L2 enables error reporting for the logged error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 44: MCA\_STATUS\_L2

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
MultiHit	0x0	1	1	1	0	0	1
Tag	0x1	0/1	0/1	0/1	0	0	1
Data	0x2	0/1	0/1	0/1	0/1	0	1
Hwa	0x3	1	1	1	0	0	1
Sdp	0x4	0/1	0/1	0/1	0/1	0	0
StateMachine	0x5	0/1	0/1	0/1	0/1	0	1
Wdt	0x6	0	0	0	0	0	1



**MSR0000\_040A...MSRC000\_2022 [L2 Machine Check Address Thread 0] (MCA::L2::MCA\_ADDR\_L2)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

MCA::L2::MCA\_ADDR\_L2 stores an address and other information associated with the error in MCA::L2::MCA\_STATUS\_L2. The register is only meaningful if MCA::L2::MCA\_STATUS\_L2[Val]=1 and MCA::L2::MCA\_STATUS\_L2[AddrV]=1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSRLEGACY; MSR0000\_040A

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_2022

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::L2::MCA_STATUS_L2. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 45: MCA\_ADDR\_L2

Error Type	Bits	Description
MultiHit	[55:52] [51:6] [5:0]	Reserved Physical Address Reserved
Tag	[55:52] [51:6] [5:0]	Reserved Physical Address Reserved
Data	[55:52] [51:6] [5:0]	Reserved Physical Address Reserved
Hwa	[31:0]	Reserved
Sdp	[55:0]	Reserved
StateMachine	[63:0]	Reserved
Wdt	[63:0]	Reserved

**MSR0000\_040B...MSRC000\_2023 [L2 Machine Check Miscellaneous 0 Thread 0] (MCA::L2::MCA\_MISC0\_L2)**

Log miscellaneous information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSRLEGACY; MSR0000\_040B

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_2023

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L2::MCA_MISC0_L2[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L2::MCA_MISC0_L2[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L2::MCA_MISC0_L2[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L2::MCA_MISC0_L2[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L2::MCA_MISC0_L2[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L2::MCA_MISC0_L2[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_2024 [L2 Machine Check Configuration Thread 0] (MCA::L2::MCA\_CONFIG\_L2)**

Reset: 0000_0000_0000_0125h.	
Controls configuration of the associated machine check bank.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst2_n[31:0]_aliasMSR; MSRC000_2024	
Bits	Description
63:41	Reserved.
40	<b>IntEn.</b> Read-write. Reset: 0. Init: BIOS,0. 1=When set, this bank will generate corrected error interrupts.
39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:35	Reserved.
34	<b>LogDeferredInMcaStat.</b> Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::L2::MCA_STATUS_L2 and MCA::L2::MCA_ADDR_L2 in addition to MCA::L2::MCA_DESTAT_L2 and MCA::L2::MCA_DEADDR_L2. 0=Only log deferred errors in MCA::L2::MCA_DESTAT_L2 and MCA::L2::MCA_DEADDR_L2. This bit does not affect logging of deferred errors in MCA::L2::MCA_SYND_L2, MCA::L2::MCA_MISC0_L2.
33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:11	Reserved.
10	<b>IntPresent.</b> Read-only, Volatile. Reset: 0. 1=This bank can be configured to trigger a corrected error interrupt using MCA::L2::MCA_CONFIG_L2[IntEn].
9	<b>McaFruTextInMca.</b> Read-write. Reset: 0. Init: BIOS,0. 1=FruText is reported McaSynd1/McaSynd2 registers
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::L2::MCA_CONFIG_L2[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::L2::MCA_CONFIG_L2[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::L2::MCA_CONFIG_L2[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::L2::MCA_CONFIG_L2[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::L2::MCA_DESTAT_L2 and MCA::L2::MCA_DEADDR_L2 are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::L2::MCA_MISC0_L2[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::L2::MCA_STATUS_L2[TCC] is present.

**MSRC000\_2025 [L2 IP Identification Thread 0] (MCA::L2::MCA\_IPID\_L2)**

Reset: 0002\_00B0\_0000\_0000h.

The MCA::L2::MCA\_IPID\_L2 register is used by software to determine what IP type and revision is associated with the MCA bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_2025

Bits	Description
63:48	<b>McaType</b> . Read-only. Reset: 0002h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi</b> . Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID</b> . Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId</b> . Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_2026 [L2 Machine Check Syndrome Thread 0] (MCA::L2::MCA\_SYND\_L2)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::L2::MCA\_STATUS\_L2 Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_2026

Bits	Description
63:32	<b>Syndrom</b> . Read-write, Volatile. Reset: Cold, 0000_0000h. Contains the syndrome, if any, associated with the error logged in MCA::L2::MCA_STATUS_L2. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::L2::MCA_SYND_L2[Length]. The Syndrome field is only valid when MCA::L2::MCA_SYND_L2[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority</b> . Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::L2::MCA_SYND_L2. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length</b> . Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::L2::MCA_SYND_L2[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::L2::MCA_SYND_L2. For example, a syndrome length of 9 means that MCA::L2::MCA_SYND_L2[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation</b> . Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 46 [MCA_SYND_L2].

Table 46: MCA\_SYND\_L2

Error Type	Bits	Description
MultiHit	[18:8] [7:0]	Index One-hot way vector
Tag	[17:14] [13:3] [2:0]	Reserved Index Way
Data	[17:17] [16] [15:5] [4] [3] [2:0]	Reserved Poison Index TopBotLoc Reserved Way
Hwa	[17:0]	Reserved
Sdp	[17:17] [16:6] [5:3]	Reserved Addr[16:6] Addr[5:3]

	[2:0]	L2Way
StateMachine	[17:2]	Reserved
	[1]	Reserved
	[0]	Reserved
Wdt	[17:0]	Reserved

**MSRC000\_2028 [L2 Machine Check Deferred Error Status Thread 0] (MCA::L2::MCA\_DESTAT\_L2)**

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_2028

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::L2::MCA_DEADDR_L2 contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::L2::MCA_SYND_L2. If MCA::L2::MCA_SYND_L2[ErrorPriority] is the same as the priority of the error in MCA::L2::MCA_STATUS_L2, then the information in MCA::L2::MCA_SYND_L2 is associated with the error in MCA::L2::MCA_DESTAT_L2.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h. MCA_DEFSTAT Register Reserved bits.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h. MCA_DEFSTAT Register Reserved bits.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::L2::MCA_ADDR_L2[ErrorAddr]. A value of 0 indicates that MCA::L2::MCA_ADDR_L2[63:0] contains a valid byte address. A value of 6 indicates that MCA::L2::MCA_ADDR_L2[63:6] contains a valid cache line address and that MCA::L2::MCA_ADDR_L2[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::L2::MCA_ADDR_L2[63:12] contain a valid 4KB memory page and that MCA::L2::MCA_ADDR_L2[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC000\_2029 [L2 Deferred Error Address Thread 0] (MCA::L2::MCA\_DEADDR\_L2)**

Read-write, Volatile. Reset: Cold,0000\_0000\_0000\_0000h.

The MCA::L2::MCA\_DEADDR\_L2 register stores the address associated with the error in MCA::L2::MCA\_DESTAT\_L2. The register is only meaningful if MCA::L2::MCA\_DESTAT\_L2[Val]=1 and MCA::L2::MCA\_DESTAT\_L2[AddrV]=1. The lowest valid bit of the address is defined by MCA::L2::MCA\_DESTAT\_L2[AddrLsb].

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_2029

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::L2::MCA_DESTAT_L2. The lowest-order valid bit of the address is specified in MCA::L2::MCA_DESTAT_L2[AddrLsb].

**MSRC001\_0402 [L2 Machine Check Control Mask Thread 0] (MCA::L2::MCA\_CTL\_MASK\_L2)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC001\_0402

Bits	Description
63:7	Reserved.
6	<b>Wdt.</b> Read-write. Reset: 0. Reserved
5	<b>StateMachine.</b> Read-write. Reset: 0. Error initiated by programmable state machine.
4	<b>Sdp.</b> Read-write. Reset: 0. SDP Read Response Parity Error
3	<b>Hwa.</b> Read-write. Reset: 0. Hardware Assert Error.
2	<b>Data.</b> Read-write. Reset: 0. L2M Data Array ECC Error.
1	<b>Tag.</b> Read-write. Reset: 0. L2M Tag or State Array ECC Error.
0	<b>MultiHit.</b> Read-write. Reset: 0. L2M Tag Multiple-Way-Hit error.

**MSRC000\_202E [L2 Machine Check Syndrome Extended Thread 0] (MCA::L2::MCA\_SYND1\_L2)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::L2::MCA\_STATUS\_L2 Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_202E

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::L2::MCA_SYND1_L2 register stores information associated with the error in MCA::L2::MCA_STATUS_L2 or MCA_DESTAT. The register is meaningful if MCA::L2::MCA_STATUS_L2[SyndV]=1. When MCA::L2::MCA_CONFIG_L2[McaFruTextInMca]=1, MCA::L2::MCA_SYND1_L2 stores ASCII FruText associated with the error.

**MSRC000\_202F [L2 Machine Check Syndrome Extended Thread 0] (MCA::L2::MCA\_SYND2\_L2)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::L2::MCA\_STATUS\_L2 Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst2\_n[31:0]\_aliasMSR; MSRC000\_202F

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::L2::MCA_SYND2_L2 register stores information associated with the error in MCA::L2::MCA_STATUS_L2 or MCA_DESTAT. The register is meaningful if MCA::L2::MCA_STATUS_L2[SyndV]=1. When MCA::L2::MCA_CONFIG_L2[McaFruTextInMca]=1, MCA::L2::MCA_SYND2_L2 stores ASCII FruText associated with the error.

## 3.2.6.4 DE

**MSR0000\_040C...MSRC000\_2030 [DE Machine Check Control Thread 0] (MCA::DE::MCA\_CTL\_DE)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::DE::MCA\_CTL\_DE register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSRLEGACY; MSR0000\_040C

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSR; MSRC000\_2030

Bits	Description
63:10	Reserved.
9	<b>HWA.</b> Read-write. Reset: 0. Hardware Assertion Error.
8	<b>OCQ.</b> Read-write. Reset: 0. Micro-op fetch queue parity error.
7	<b>UcSeq.</b> Read-write. Reset: 0. Patch RAM sequencer parity error.
6	<b>UcDat.</b> Read-write. Reset: 0. Patch RAM data parity error.
5	<b>Faq.</b> Read-write. Reset: 0. Fetch address FIFO parity error.
4	<b>Idq.</b> Read-write. Reset: 0. Instruction dispatch queue parity error.
3	<b>UopQ.</b> Read-write. Reset: 0. Micro-op queue parity error.
2	<b>Ibq.</b> Read-write. Reset: 0. Instruction buffer parity error.
1	<b>OcDat.</b> Read-write. Reset: 0. Micro-op cache data parity error.
0	<b>OcTag.</b> Read-write. Reset: 0. Micro-op cache tag parity error.



**MSR0000\_040D...MSRC000\_2031 [DE Machine Check Status Thread 0] (MCA::DE::MCA\_STATUS\_DE)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSRLEGACY; MSR0000\_040D

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSR; MSRC000\_2031

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::DE::MCA_CTL_DE. This bit is a copy of bit in MCA::DE::MCA_CTL_DE for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::DE::MCA_MISC0_DE. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::DE::MCA_ADDR_DE contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::DE::MCA_STATUS_DE[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::DE::MCA_SYND_DE. If MCA::DE::MCA_SYND_DE[ErrorPriority] is the same as the priority of the error in MCA::DE::MCA_STATUS_DE, then the information in MCA::DE::MCA_SYND_DE is associated with the error in MCA::DE::MCA_STATUS_DE. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.



44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::DE::MCA_ADDR_DE[ErrorAddr]. A value of 0 indicates that MCA::DE::MCA_ADDR_DE[63:0] contains a valid byte address. A value of 6 indicates that MCA::DE::MCA_ADDR_DE[63:6] contains a valid cache line address and that MCA::DE::MCA_ADDR_DE[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::DE::MCA_ADDR_DE[63:12] contain a valid 4KB memory page and that MCA::DE::MCA_ADDR_DE[11:0] should be ignored by error handling software. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::DE::MCA_CTL_DE enables error reporting for the logged error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 47: MCA\_STATUS\_DE

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
OcTag	0x0	0	0	0	0	0	0
OcDat	0x1	0	0	0	0	0	0
Ibq	0x2	1	1	1	0	0	0
UopQ	0x3	1	1	1	0	0	0
Idq	0x4	1	1	1	0	0	0
Faq	0x5	1	1	1	0	0	0
UcDat	0x6	1	1	1	0	0	0
UcSeq	0x7	1	1	1	0	0	0
OCQ	0x8	0	0	0	0	0	0
HWA	0x9	1	1	1	0	0	0
Reserved	0xa	0	0	0	0	0	1

**MSR0000\_040E...MSRC000\_2032 [DE Machine Check Address Thread 0] (MCA::DE::MCA\_ADDR\_DE)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

MCA::DE::MCA\_ADDR\_DE stores an address and other information associated with the error in MCA::DE::MCA\_STATUS\_DE. The register is only meaningful if MCA::DE::MCA\_STATUS\_DE[Val]=1 and MCA::DE::MCA\_STATUS\_DE[AddrV]=1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSRLEGACY; MSR0000\_040E

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSR; MSRC000\_2032

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::DE::MCA_STATUS_DE. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 48: MCA\_ADDR\_DE

Error Type	Bits	Description
OcTag	[63:0]	Reserved
OcDat	[63:0]	Reserved
Ibq	[63:0]	Reserved
UopQ	[63:0]	Reserved
Idq	[63:0]	Reserved
Faq	[63:0]	Reserved
UcDat	[63:0]	Reserved
UcSeq	[63:0]	Reserved
OCQ	[63:0]	Reserved
HWA	[63:0]	Reserved
Reserved	[63:0]	Reserved

**MSR0000\_040F...MSRC000\_2033 [DE Machine Check Miscellaneous 0 Thread 0]  
(MCA::DE::MCA\_MISC0\_DE)**

Log miscellaneous information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSRLEGACY; MSR0000\_040F

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSR; MSRC000\_2033

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::DE::MCA_MISC0_DE[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::DE::MCA_MISC0_DE[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::DE::MCA_MISC0_DE[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::DE::MCA_MISC0_DE[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::DE::MCA_MISC0_DE[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::DE::MCA_MISC0_DE[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_2034 [DE Machine Check Configuration Thread 0] (MCA::DE::MCA\_CONFIG\_DE)**

Reset: 0000_0000_0000_0121h.	
Controls configuration of the associated machine check bank.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst3_n[31:0]_aliasMSR; MSRC000_2034	
Bits	Description
63:41	Reserved.
40	<b>IntEn.</b> Read-write. Reset: 0. Init: BIOS,0. 1=When set, this bank will generate corrected error interrupts.
39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:11	Reserved.
10	<b>IntPresent.</b> Read-only, Volatile. Reset: 0. 1=This bank can be configured to trigger a corrected error interrupt using MCA::DE::MCA_CONFIG_DE[IntEn].
9	<b>McaFruTextInMca.</b> Read-write. Reset: 0. Init: BIOS,0. 1=FruText is reported McaSynd1/McaSynd2 registers
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::DE::MCA_CONFIG_DE[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::DE::MCA_CONFIG_DE[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::DE::MCA_CONFIG_DE[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 0. 1=Deferred errors are supported in this MCA bank, and the LogDeferredInMcaStat field in this register controls the logging behavior of these errors. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::DE::MCA_MISC0_DE[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::DE::MCA_STATUS_DE[TCC] is present.

**MSRC000\_2035 [DE IP Identification Thread 0] (MCA::DE::MCA\_IPID\_DE)**

Reset: 0003_00B0_0000_0000h.	
The MCA::DE::MCA_IPID_DE register is used by software to determine what IP type and revision is associated with the MCA bank.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst3_n[31:0]_aliasMSR; MSRC000_2035	
Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0003h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_2036 [DE Machine Check Syndrome Thread 0] (MCA::DE::MCA\_SYND\_DE)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::DE::MCA\_STATUS\_DE Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSR; MSRC000\_2036

Bits	Description
63:32	<b>Syndrom</b> . Read-write, Volatile. Reset: Cold, 0000_0000h. Contains the syndrome, if any, associated with the error logged in MCA::DE::MCA_STATUS_DE. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::DE::MCA_SYND_DE[Length]. The Syndrome field is only valid when MCA::DE::MCA_SYND_DE[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority</b> . Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::DE::MCA_SYND_DE. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length</b> . Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::DE::MCA_SYND_DE[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::DE::MCA_SYND_DE. For example, a syndrome length of 9 means that MCA::DE::MCA_SYND_DE[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation</b> . Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 49 [MCA_SYND_DE].

Table 49: MCA\_SYND\_DE

Error Type	Bits	Description
OcTag	[17:10] [9:6] [5:0]	Reserved Way Index
OcDat	[17:10] [9:6] [5:0]	Reserved Way Index
Ibq	[17:0]	Reserved
UopQ	[17:0]	Reserved
Idq	[17:0]	Reserved
Faq	[17:0]	Reserved
UcDat	[17:0]	Reserved
UcSeq	[17:0]	Reserved
OCQ	[17:0]	Reserved
HWA	[17:6] [5:0]	Reserved Reserved
Reserved	[17:0]	Reserved

**MSRC001\_0403 [DE Machine Check Control Mask Thread 0] (MCA::DE::MCA\_CTL\_MASK\_DE)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSR; MSRC001\_0403

Bits	Description
63:10	Reserved.
9	<b>HWA.</b> Read-write. Reset: 0. Hardware Assertion Error.
8	<b>OCQ.</b> Read-write. Reset: 0. Micro-op fetch queue parity error.
7	<b>UcSeq.</b> Read-write. Reset: 0. Patch RAM sequencer parity error.
6	<b>UcDat.</b> Read-write. Reset: 0. Patch RAM data parity error.
5	<b>Faq.</b> Read-write. Reset: 0. Fetch address FIFO parity error.
4	<b>Idq.</b> Read-write. Reset: 0. Instruction dispatch queue parity error.
3	<b>UopQ.</b> Read-write. Reset: 0. Micro-op queue parity error.
2	<b>Ibq.</b> Read-write. Reset: 0. Instruction buffer parity error.
1	<b>OcDat.</b> Read-write. Reset: 0. Micro-op cache data parity error.
0	<b>OcTag.</b> Read-write. Reset: 0. Micro-op cache tag parity error.

**MSRC000\_203E [DE Machine Check Syndrome Extended Thread 0] (MCA::DE::MCA\_SYND1\_DE)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::DE::MCA\_STATUS\_DE Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSR; MSRC000\_203E

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::DE::MCA_SYND1_DE register stores information associated with the error in MCA::DE::MCA_STATUS_DE or MCA_DESTAT. The register is meaningful if MCA::DE::MCA_STATUS_DE[SyndV]=1. When MCA::DE::MCA_CONFIG_DE[McaFruTextInMca]=1, MCA::DE::MCA_SYND1_DE stores ASCII FruText associated with the error.

**MSRC000\_203F [DE Machine Check Syndrome Extended Thread 0] (MCA::DE::MCA\_SYND2\_DE)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::DE::MCA\_STATUS\_DE Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst3\_n[31:0]\_aliasMSR; MSRC000\_203F

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::DE::MCA_SYND2_DE register stores information associated with the error in MCA::DE::MCA_STATUS_DE or MCA_DESTAT. The register is meaningful if MCA::DE::MCA_STATUS_DE[SyndV]=1. When MCA::DE::MCA_CONFIG_DE[McaFruTextInMca]=1, MCA::DE::MCA_SYND2_DE stores ASCII FruText associated with the error.

## 3.2.6.5 EX

**MSR0000\_0414...MSRC000\_2050 [EX Machine Check Control Thread 0] (MCA::EX::MCA\_CTL\_EX)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::EX::MCA\_CTL\_EX register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0414

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSR; MSRC000\_2050

Bits	Description
63:14	Reserved.
13	<b>RETMAP</b> . Read-write. Reset: 0. Retire Map parity error.
12	<b>SPECMAP</b> . Read-write. Reset: 0. PRN/FRN freelist parity error.
11	<b>HWA</b> . Read-write. Reset: 0. Hardware Assertion error.
10	<b>BBQ</b> . Read-write. Reset: 0. Branch buffer queue parity error.
9	<b>SQ</b> . Read-write. Reset: 0. EXTID parity error.
8	<b>STATQ</b> . Read-write. Reset: 0. Retire status queue parity error.
7	<b>RETDISP</b> . Read-write. Reset: 0. Retire dispatch queue parity error.
6	<b>CHKPTQ</b> . Read-write. Reset: 0. CHKPTQ. Checkpoint queue parity error.
5	<b>PLDAL</b> . Read-write. Reset: 0. EX payload parity error.
4	<b>PLDAG</b> . Read-write. Reset: 0. Address generator payload parity error.
3	<b>IDRF</b> . Read-write. Reset: 0. Immediate displacement register file parity error.
2	<b>FRF</b> . Read-write. Reset: 0. Flag register file parity error.
1	<b>PRF</b> . Read-write. Reset: 0. Physical register file parity error.
0	<b>WDT</b> . Read-write. Reset: 0. Watchdog Timeout error.



**MSR0000\_0415...MSRC000\_2051 [EX Machine Check Status Thread 0] (MCA::EX::MCA\_STATUS\_EX)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0415

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSR; MSRC000\_2051

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::EX::MCA_CTL_EX. This bit is a copy of bit in MCA::EX::MCA_CTL_EX for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::EX::MCA_MISC0_EX. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::EX::MCA_ADDR_EX contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::EX::MCA_STATUS_EX[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::EX::MCA_SYND_EX. If MCA::EX::MCA_SYND_EX[ErrorPriority] is the same as the priority of the error in MCA::EX::MCA_STATUS_EX, then the information in MCA::EX::MCA_SYND_EX is associated with the error in MCA::EX::MCA_STATUS_EX. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.



44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::EX::MCA_ADDR_EX[ErrorAddr]. A value of 0 indicates that MCA::EX::MCA_ADDR_EX[63:0] contains a valid byte address. A value of 6 indicates that MCA::EX::MCA_ADDR_EX[63:6] contains a valid cache line address and that MCA::EX::MCA_ADDR_EX[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::EX::MCA_ADDR_EX[63:12] contain a valid 4KB memory page and that MCA::EX::MCA_ADDR_EX[11:0] should be ignored by error handling software. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::EX::MCA_CTL_EX enables error reporting for the logged error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 50: MCA\_STATUS\_EX

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
WDT	0x0	1	1	1	0	0	1
PRF	0x1	1	1	1	0	0	1
FRF	0x2	1	1	1	0	0	1
IDRF	0x3	1	1	1	0	0	1
PLDAG	0x4	1	1	1	0	0	0
PLDAL	0x5	1	1	1	0	0	1
CHKPTQ	0x6	1	1	1	0	0	1
RETDISP	0x7	1	1	1	0	0	1
STATQ	0x8	1	1	1	0	0	0
SQ	0x9	1	1	1	0	0	1
BBQ	0xa	1	1	1	0	0	1
HWA	0xb	1	1	1	0	0	1

SPECMAP	0xc	1	1	1	0	0	1
RETMAP	0xd	1	1	1	0	0	0
Reserved	0xe	0	0	0	0	0	1

**MSR0000\_0416...MSRC000\_2052 [EX Machine Check Address Thread 0] (MCA::EX::MCA\_ADDR\_EX)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

MCA::EX::MCA\_ADDR\_EX stores an address and other information associated with the error in MCA::EX::MCA\_STATUS\_EX. The register is only meaningful if MCA::EX::MCA\_STATUS\_EX[Val]=1 and MCA::EX::MCA\_STATUS\_EX[AddrV]=1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0416

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSR; MSRC000\_2052

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::EX::MCA_STATUS_EX. For physical addresses, the most significant bit is given by Core::X86::CpuId::LongModeInfo[PhysAddrSize].

Table 51: MCA\_ADDR\_EX

Error Type	Bits	Description
WDT	[56:0]	RIP of thread triggering the watchdog timeout
PRF	[63:0]	Reserved
FRF	[63:0]	Reserved
IDRF	[63:0]	Reserved
PLDAG	[63:0]	Reserved
PLDAL	[63:0]	Reserved
CHKPTQ	[63:0]	Reserved
RETDISP	[63:0]	Reserved
STATQ	[63:0]	Reserved
SQ	[63:0]	Reserved
BBQ	[63:0]	Reserved
HWA	[63:0]	Reserved
SPECMAP	[63:0]	Reserved
RETMAP	[56:0]	Reserved
Reserved	[63:0]	Reserved

**MSR0000\_0417...MSRC000\_2053 [EX Machine Check Miscellaneous 0 Thread 0]  
(MCA::EX::MCA\_MISC0\_EX)**

Log miscellaneous information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0417

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSR; MSRC000\_2053

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msrr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_2054 [EX Machine Check Configuration Thread 0] (MCA::EX::MCA\_CONFIG\_EX)**

Reset: 0000_0000_0000_0121h.	
Controls configuration of the associated machine check bank.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst5_n[31:0]_aliasMSR; MSRC000_2054	
Bits	Description
63:41	Reserved.
40	<b>IntEn.</b> Read-write. Reset: 0. Init: BIOS,0. 1=When set, this bank will generate corrected error interrupts.
39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:11	Reserved.
10	<b>IntPresent.</b> Read-only, Volatile. Reset: 0. 1=This bank can be configured to trigger a corrected error interrupt using MCA::EX::MCA_CONFIG_EX[IntEn].
9	<b>McaFruTextInMca.</b> Read-write. Reset: 0. Init: BIOS,0. 1=FruText is reported McaSynd1/McaSynd2 registers
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::EX::MCA_CONFIG_EX[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::EX::MCA_CONFIG_EX[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::EX::MCA_CONFIG_EX[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 0. 1=Deferred errors are supported in this MCA bank, and the LogDeferredInMcaStat field in this register controls the logging behavior of these errors. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::EX::MCA_MISC0_EX[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::EX::MCA_STATUS_EX[TCC] is present.

**MSRC000\_2055 [EX IP Identification Thread 0] (MCA::EX::MCA\_IPID\_EX)**

Reset: 0005_00B0_0000_0000h.	
The MCA::EX::MCA_IPID_EX register is used by software to determine what IP type and revision is associated with the MCA bank.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst5_n[31:0]_aliasMSR; MSRC000_2055	
Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0005h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_2056 [EX Machine Check Syndrome Thread 0] (MCA::EX::MCA\_SYND\_EX)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::EX::MCA\_STATUS\_EX Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSR; MSRC000\_2056

Bits	Description
63:32	<b>Syndrone.</b> Read-write, Volatile. Reset: Cold, 0000_0000h. Contains the syndrome, if any, associated with the error logged in MCA::EX::MCA_STATUS_EX. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::EX::MCA_SYND_EX[Length]. The Syndrome field is only valid when MCA::EX::MCA_SYND_EX[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority.</b> Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::EX::MCA_SYND_EX. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length.</b> Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::EX::MCA_SYND_EX[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::EX::MCA_SYND_EX. For example, a syndrome length of 9 means that MCA::EX::MCA_SYND_EX[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation.</b> Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 52 [MCA_SYND_EX].

Table 52: MCA\_SYND\_EX

Error Type	Bits	Description
WDT	[17:0]	Reserved
PRF	[17:0]	Reserved
FRF	[17:8] [7:0]	0 FRF Index
IDRF	[17:8] [7:7] [6:0]	0 1: ALSQ, 0: AGSQ Index
PLDAG	[17:8] [7:7] [6:0]	0 1: AGSQ PLD, 0: AGSQ EPLD Index
PLDAL	[17:8] [7:7] [6:0]	0 1: ALSQ PLD, 0: ALSQ EPLD Index
CHKPTQ	[17:4] [3] [2] [1] [0]	0 Thread 1 Parity Error Thread 0 Parity Error Thread 1 Flush Parity Error Thread 0 Flush Parity Error
RETDISP	[17:2] [1] [0]	0 Thread 1 Parity Error Thread 0 Parity Error
STATQ	[17:0]	Reserved
SQ	[17:6] [5:0]	0 EXTID
BBQ	[17:3] [2] [1] [0]	0 RIP Parity Error FIP Parity Error LBF Parity Error

HWA	[17:6] [5:0]	Reserved Reserved
SPECMAP	[17:2] [1] [0]	0 Reserved Reserved
RETMAP	[17:0]	Reserved
Reserved	[17:0]	Reserved

**MSRC001\_0405 [EX Machine Check Control Mask Thread 0] (MCA::EX::MCA\_CTL\_MASK\_EX)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSR; MSRC001\_0405

Bits	Description
63:14	Reserved.
13	<b>RETMAP</b> . Read-write. Reset: 0. Retire Map parity error.
12	<b>SPECMAP</b> . Read-write. Reset: 0. PRN/FRN freelist parity error.
11	<b>HWA</b> . Read-write. Reset: 0. Hardware Assertion error.
10	<b>BBQ</b> . Read-write. Reset: 0. Branch buffer queue parity error.
9	<b>SQ</b> . Read-write. Reset: 0. EXTID parity error.
8	<b>STATQ</b> . Read-write. Reset: 0. Retire status queue parity error.
7	<b>RETDISP</b> . Read-write. Reset: 0. Retire dispatch queue parity error.
6	<b>CHKPTQ</b> . Read-write. Reset: 0. CHKPTQ. Checkpoint queue parity error.
5	<b>PLDAL</b> . Read-write. Reset: 0. EX payload parity error.
4	<b>PLDAG</b> . Read-write. Reset: 0. Address generator payload parity error.
3	<b>IDRF</b> . Read-write. Reset: 0. Immediate displacement register file parity error.
2	<b>FRF</b> . Read-write. Reset: 0. Flag register file parity error.
1	<b>PRF</b> . Read-write. Reset: 0. Physical register file parity error.
0	<b>WDT</b> . Read-write. Reset: 0. Watchdog Timeout error.

**MSRC000\_205E [EX Machine Check Syndrome Extended Thread 0] (MCA::EX::MCA\_SYND1\_EX)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::EX::MCA\_STATUS\_EX Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSR; MSRC000\_205E

Bits	Description
63:0	<b>Syndrome</b> . Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::EX::MCA_SYND1_EX register stores information associated with the error in MCA::EX::MCA_STATUS_EX or MCA_DESTAT. The register is meaningful if MCA::EX::MCA_STATUS_EX[SyndV]=1. When MCA::EX::MCA_CONFIG_EX[McaFruTextInMca]=1, MCA::EX::MCA_SYND1_EX stores ASCII FruText associated with the error.

**MSRC000\_205F [EX Machine Check Syndrome Extended Thread 0] (MCA::EX::MCA\_SYND2\_EX)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::EX::MCA\_STATUS\_EX Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst5\_n[31:0]\_aliasMSR; MSRC000\_205F

Bits	Description
63:0	<b>Syndrome</b> . Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::EX::MCA_SYND2_EX register stores information associated with the error in MCA::EX::MCA_STATUS_EX or MCA_DESTAT. The register is meaningful if MCA::EX::MCA_STATUS_EX[SyndV]=1. When MCA::EX::MCA_CONFIG_EX[McaFruTextInMca]=1, MCA::EX::MCA_SYND2_EX stores ASCII FruText associated with the error.

**3.2.6.6 FP****MSR0000\_0418...MSRC000\_2060 [FP Machine Check Control Thread 0] (MCA::FP::MCA\_CTL\_FP)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::FP::MCA\_CTL\_FP register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0418

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSR; MSRC000\_2060

Bits	Description
63:8	Reserved.
7	<b>KRF</b> . Read-write. Reset: 0. Physical K mask register file (KRF) parity error.
6	<b>HWA</b> . Read-write. Reset: 0. Hardware assertion.
5	<b>SRF</b> . Read-write. Reset: 0. Status register file (SRF) parity error.
4	<b>RQ</b> . Read-write. Reset: 0. Retire queue (RQ) parity error.
3	<b>NSQ</b> . Read-write. Reset: 0. NSQ parity error.
2	<b>SCH</b> . Read-write. Reset: 0. Schedule queue parity error.
1	<b>FL</b> . Read-write. Reset: 0. Freelist (FL) parity error.
0	<b>PRF</b> . Read-write. Reset: 0. Physical register file (PRF) parity error.



**MSR0000\_0419...MSRC000\_2061 [FP Machine Check Status Thread 0] (MCA::FP::MCA\_STATUS\_FP)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSRLEGACY; MSR0000\_0419

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSR; MSRC000\_2061

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::FP::MCA_CTL_FP. This bit is a copy of bit in MCA::FP::MCA_CTL_FP for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::FP::MCA_MISC0_FP. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::FP::MCA_ADDR_FP contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::FP::MCA_STATUS_FP[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::FP::MCA_SYND_FP. If MCA::FP::MCA_SYND_FP[ErrorPriority] is the same as the priority of the error in MCA::FP::MCA_STATUS_FP, then the information in MCA::FP::MCA_SYND_FP is associated with the error in MCA::FP::MCA_STATUS_FP. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.



44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::FP::MCA_ADDR_FP[ErrorAddr]. A value of 0 indicates that MCA::FP::MCA_ADDR_FP[63:0] contains a valid byte address. A value of 6 indicates that MCA::FP::MCA_ADDR_FP[63:6] contains a valid cache line address and that MCA::FP::MCA_ADDR_FP[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::FP::MCA_ADDR_FP[63:12] contain a valid 4KB memory page and that MCA::FP::MCA_ADDR_FP[11:0] should be ignored by error handling software. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::FP::MCA_CTL_FP enables error reporting for the logged error. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 53: MCA\_STATUS\_FP

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
PRF	0x0	1	1	1	0	0	0
FL	0x1	1	1	1	0	0	0
SCH	0x2	1	1	1	0	0	0
NSQ	0x3	1	1	1	0	0	0
RQ	0x4	1	1	1	0	0	0
SRF	0x5	1	1	1	0	0	0
HWA	0x6	1	1	1	0	0	0
KRF	0x7	1	1	1	0	0	0

**MSR0000\_041A...MSRC000\_2062 [FP Machine Check Address Thread 0] (MCA::FP::MCA\_ADDR\_FP)**

Read-only. Reset: Cold,0000\_0000\_0000\_0000h.

MCA::FP::MCA\_ADDR\_FP stores an address and other information associated with the error in MCA::FP::MCA\_STATUS\_FP. The register is only meaningful if MCA::FP::MCA\_STATUS\_FP[Val]=1 and MCA::FP::MCA\_STATUS\_FP[AddrV]=1.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSRLEGACY; MSR0000\_041A

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSR; MSRC000\_2062

Bits	Description
63:0	<b>ErrorAddr.</b> Read-only. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::FP::MCA_STATUS_FP.

Table 54: MCA\_ADDR\_FP

Error Type	Bits	Description
PRF	[55:0]	Reserved
FL	[55:0]	Reserved
SCH	[55:0]	Reserved
NSQ	[55:0]	Reserved
RQ	[55:0]	Reserved
SRF	[55:0]	Reserved
HWA	[55:0]	Reserved
KRF	[55:0]	Reserved

**MSR0000\_041B...MSRC000\_2063 [FP Machine Check Miscellaneous 0 Thread 0] (MCA::FP::MCA\_MISC0\_FP)**

Log miscellaneous information associated with errors.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSRLEGACY; MSR0000\_041B

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSR; MSRC000\_2063

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::FP::MCA_MISC0_FP[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::FP::MCA_MISC0_FP[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::FP::MCA_MISC0_FP[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrlf is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::FP::MCA_MISC0_FP[Locked]) ? Read-write : Read-only.
48	<b>Ovrlf.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::FP::MCA_MISC0_FP[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::FP::MCA_MISC0_FP[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_2064 [FP Machine Check Configuration Thread 0] (MCA::FP::MCA\_CONFIG\_FP)**

Reset: 0000\_0002\_0000\_0121h.

Controls configuration of the associated machine check bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSR; MSRC000\_2064

Bits	Description
63:41	Reserved.
40	<b>IntEn.</b> Read-write. Reset: 0. Init: BIOS,0. 1=When set, this bank will generate corrected error interrupts.
39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:11	Reserved.
10	<b>IntPresent.</b> Read-only, Volatile. Reset: 0. 1=This bank can be configured to trigger a corrected error interrupt using MCA::FP::MCA_CONFIG_FP[IntEn].
9	<b>McaFruTextInMca.</b> Read-write. Reset: 0. Init: BIOS,0. 1=FruText is reported McaSynd1/McaSynd2 registers
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::FP::MCA_CONFIG_FP[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::FP::MCA_CONFIG_FP[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::FP::MCA_CONFIG_FP[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 0. 1=Deferred errors are supported in this MCA bank, and the LogDeferredInMcaStat field in this register controls the logging behavior of these errors. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::FP::MCA_MISC0_FP[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::FP::MCA_STATUS_FP[TCC] is present.

**MSRC000\_2065 [FP IP Identification Thread 0] (MCA::FP::MCA\_IPID\_FP)**

Reset: 0006\_00B0\_0000\_0000h.

The MCA::FP::MCA\_IPID\_FP register is used by software to determine what IP type and revision is associated with the MCA bank.

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSR; MSRC000\_2065

Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0006h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_2066 [FP Machine Check Syndrome Thread 0] (MCA::FP::MCA\_SYND\_FP)**

Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h.	
Logs physical location information associated with error in MCA::FP::MCA_STATUS_FP Thread 0	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst6_n[31:0]_aliasMSR; MSRC000_2066	
Bits	Description
63:27	Reserved.
26:24	<b>ErrorPriority.</b> Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::FP::MCA_SYND_FP. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length.</b> Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of any syndromes logged. Only meaningful if the Syndrome field exists in this register.
17:0	<b>ErrorInformation.</b> Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 55 [MCA_SYND_FP].

Table 55: MCA\_SYND\_FP

Error Type	Bits	Description
PRF	[17:0]	Reserved
FL	[17:0]	Reserved
SCH	[17:0]	Reserved
NSQ	[17:0]	Reserved
RQ	[17:0]	Reserved
SRF	[17:0]	Reserved
HWA	[17:0]	Reserved
KRF	[17:0]	Reserved

**MSRC001\_0406 [FP Machine Check Control Mask Thread 0] (MCA::FP::MCA\_CTL\_MASK\_FP)**

Read-write. Reset: 0000_0000_0000_0000h.	
Inhibit detection of an error source.	
_ccd[1:0]_lthree0_core[7:0]_thread[1:0]_inst6_n[31:0]_aliasMSR; MSRC001_0406	
Bits	Description
63:8	Reserved.
7	<b>KRF.</b> Read-write. Reset: 0. Physical K mask register file (KRF) parity error.
6	<b>HWA.</b> Read-write. Reset: 0. Hardware assertion.
5	<b>SRF.</b> Read-write. Reset: 0. Status register file (SRF) parity error.
4	<b>RQ.</b> Read-write. Reset: 0. Retire queue (RQ) parity error.
3	<b>NSQ.</b> Read-write. Reset: 0. NSQ parity error.
2	<b>SCH.</b> Read-write. Reset: 0. Schedule queue parity error.
1	<b>FL.</b> Read-write. Reset: 0. Freelist (FL) parity error.
0	<b>PRF.</b> Read-write. Reset: 0. Physical register file (PRF) parity error.

**MSRC000\_206E [FP Machine Check Syndrome Extended Thread 0] (MCA::FP::MCA\_SYND1\_FP)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::FP::MCA\_STATUS\_FP Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSR; MSRC000\_206E

Bits	Description
63:0	<b>Syndromes.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::FP::MCA_SYND1_FP register stores information associated with the error in MCA::FP::MCA_STATUS_FP or MCA_DESTAT. The register is meaningful if MCA::FP::MCA_STATUS_FP[SyndV]=1. When MCA::FP::MCA_CONFIG_FP[McaFruTextInMca]=1, MCA::FP::MCA_SYND1_FP stores ASCII FruText associated with the error.

**MSRC000\_206F [FP Machine Check Syndrome Extended Thread 0] (MCA::FP::MCA\_SYND2\_FP)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::FP::MCA\_STATUS\_FP Thread 0

\_ccd[1:0]\_lthree0\_core[7:0]\_thread[1:0]\_inst6\_n[31:0]\_aliasMSR; MSRC000\_206F

Bits	Description
63:0	<b>Syndromes.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::FP::MCA_SYND2_FP register stores information associated with the error in MCA::FP::MCA_STATUS_FP or MCA_DESTAT. The register is meaningful if MCA::FP::MCA_STATUS_FP[SyndV]=1. When MCA::FP::MCA_CONFIG_FP[McaFruTextInMca]=1, MCA::FP::MCA_SYND2_FP stores ASCII FruText associated with the error.

## 3.2.6.7 L3

**MSR0000\_041C...MSRC000\_20E0 [L3 Machine Check Control] (MCA::L3::MCA\_CTL\_L3)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::L3::MCA\_CTL\_L3 register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSRLEGACY; MSR0000\_041C

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSRLEGACY; MSR0000\_0420

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSRLEGACY; MSR0000\_0424

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSRLEGACY; MSR0000\_0428

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSRLEGACY; MSR0000\_042C

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSRLEGACY; MSR0000\_0430

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSRLEGACY; MSR0000\_0434

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSRLEGACY; MSR0000\_0438

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2070

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2080

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2090

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A0

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B0

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C0

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D0

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E0

Bits	Description
63:10	Reserved.
9	<b>DsmMce.</b> Read-write. Reset: 0. Machine check error initiated by DSM action
8	<b>XiWcbParityPoison.</b> Read-write. Reset: 0. Xi Wcb Parity Poison Creation Event
7	<b>Hwa.</b> Read-write. Reset: 0. L3 Hardware Assertion.
6	<b>XiVictimQueue.</b> Read-write. Reset: 0. L3 Victim Queue Data Fabric Error.
5	<b>SdpParity.</b> Read-write. Reset: 0. SDP Parity Error from XI.
4	<b>DataArray.</b> Read-write. Reset: 0. L3M Data ECC Error.
3	<b>MultiHitTag.</b> Read-write. Reset: 0. L3M Tag Multi-way-hit Error.
2	<b>Tag.</b> Read-write. Reset: 0. L3M Tag ECC Error.
1	<b>MultiHitShadowTag.</b> Read-write. Reset: 0. Shadow Tag Macro Multi-way-hit Error.
0	<b>ShadowTag.</b> Read-write. Reset: 0. Shadow Tag Macro ECC Error.



**MSR0000\_041D...MSRC000\_20E1 [L3 Machine Check Status] (MCA::L3::MCA\_STATUS\_L3)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSRLEGACY; MSR0000\_041D  
 \_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSRLEGACY; MSR0000\_0421  
 \_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSRLEGACY; MSR0000\_0425  
 \_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSRLEGACY; MSR0000\_0429  
 \_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSRLEGACY; MSR0000\_042D  
 \_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSRLEGACY; MSR0000\_0431  
 \_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSRLEGACY; MSR0000\_0435  
 \_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSRLEGACY; MSR0000\_0439  
 \_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2071  
 \_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2081  
 \_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2091  
 \_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A1  
 \_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B1  
 \_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C1  
 \_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D1  
 \_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E1

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::L3::MCA_CTL_L3. This bit is a copy of bit in MCA::L3::MCA_CTL_L3 for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::L3::MCA_MISC0_L3. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::L3::MCA_ADDR_L3 contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::L3::MCA_STATUS_L3[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.



53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::L3::MCA_SYND_L3. If MCA::L3::MCA_SYND_L3[ErrorPriority] is the same as the priority of the error in MCA::L3::MCA_STATUS_L3, then the information in MCA::L3::MCA_SYND_L3 is associated with the error in MCA::L3::MCA_STATUS_L3. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::L3::MCA_ADDR_L3[ErrorAddr]. A value of 0 indicates that MCA::L3::MCA_ADDR_L3[63:0] contains a valid byte address. A value of 6 indicates that MCA::L3::MCA_ADDR_L3[63:6] contains a valid cache line address and that MCA::L3::MCA_ADDR_L3[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::L3::MCA_ADDR_L3[63:12] contain a valid 4KB memory page and that MCA::L3::MCA_ADDR_L3[11:0] should be ignored by error handling software. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::L3::MCA_CTL_L3 enables error reporting for the logged error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 56: MCA\_STATUS\_L3

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
ShadowTag	0x0	0/1	0/1	0/1	0	0	1
MultiHitShadowTag	0x1	1	1	1	0	0	1
Tag	0x2	0/1	0/1	0/1	0	0	1
MultiHitTag	0x3	1	1	1	0	0	1
DataArray	0x4	0/1	0/1	0/1	0/1	0	1
SdpParity	0x5	1	1	1	0	0	1
XiVictimQueue	0x6	1	1	1	0	0	1
Hwa	0x7	1	1	1	0	0	1
XiWcbParityPoison	0x8	0	0	0	1	0	1
DsmMce	0x9	0/1	0/1	0/1	0/1	0	0

**MSR0000\_041E...MSRC000\_20E2 [L3 Machine Check Address] (MCA::L3::MCA\_ADDR\_L3)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

MCA::L3::MCA\_ADDR\_L3 stores an address and other information associated with the error in MCA::L3::MCA\_STATUS\_L3. The register is only meaningful if MCA::L3::MCA\_STATUS\_L3[Val]=1 and MCA::L3::MCA\_STATUS\_L3[AddrV]=1.

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSRLEGACY; MSR0000\_041E

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSRLEGACY; MSR0000\_0422

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSRLEGACY; MSR0000\_0426

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSRLEGACY; MSR0000\_042A

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSRLEGACY; MSR0000\_042E

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSRLEGACY; MSR0000\_0432

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSRLEGACY; MSR0000\_0436

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSRLEGACY; MSR0000\_043A

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2072

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2082

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2092

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A2

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B2

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C2

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D2

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E2

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::L3::MCA_STATUS_L3. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 57: MCA\_ADDR\_L3

Error Type	Bits	Description
ShadowTag	[55:16] [15:0]	Reserved 16'b{7'b{Index}, 3'b{Slice}, 6'b{0}}
MultiHitShadowTag	[55:16] [15:0]	Reserved 16'b{7'b{Index}, 3'b{Slice}, 6'b{0}}
Tag	[55:21] [20:0]	Reserved 21'b{1'b{L3MIF}, 7'b{Index}, 4'b{Bank[2:0]}, 3'b{Slice}, 6'b{0}}
MultiHitTag	[55:21]	Reserved

	[21:0]	21'b{1'b{L3MIF}, 7'b{Index}, 4'b{Bank[3:0]}, 3'b{Slice}, 6'b{0}}
dataArray	[55:52] [51:0]	Reserved Physical Address
SdpParity	[55:52] [51:0]	Reserved Physical Address
XiVictimQueue	[55:52] [51:0]	Reserved Physical Address
Hwa	[63:46] [45:0]	Reserved Reserved
XiWcbParityPoison	[55:52] [51:0]	Reserved Physical Address
DsmMce	[64:0]	Reserved

**MSR0000\_041F...MSRC000\_20E3 [L3 Machine Check Miscellaneous 0] (MCA::L3::MCA\_MISC0\_L3)**

Log miscellaneous information associated with errors.

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSRLEGACY; MSR0000\_041F  
 \_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSRLEGACY; MSR0000\_0423  
 \_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSRLEGACY; MSR0000\_0427  
 \_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSRLEGACY; MSR0000\_042B  
 \_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSRLEGACY; MSR0000\_042F  
 \_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSRLEGACY; MSR0000\_0433  
 \_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSRLEGACY; MSR0000\_0437  
 \_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSRLEGACY; MSR0000\_043B  
 \_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2073  
 \_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2083  
 \_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2093  
 \_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A3  
 \_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B3  
 \_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C3  
 \_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D3  
 \_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E3

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L3::MCA_MISC0_L3[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L3::MCA_MISC0_L3[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L3::MCA_MISC0_L3[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L3::MCA_MISC0_L3[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L3::MCA_MISC0_L3[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported.

	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::L3::MCA_MISC0_L3[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_20[7...E]4 [L3 Machine Check Configuration] (MCA::L3::MCA\_CONFIG\_L3)**

Reset: 0000\_0000\_0000\_0125h.

Controls configuration of the associated machine check bank.

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2074

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2084

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2094

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A4

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B4

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C4

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D4

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E4

Bits	Description
63:41	Reserved.
40	<b>IntEn.</b> Read-write. Reset: 0. Init: BIOS,0. 1=When set, this bank will generate corrected error interrupts.
39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:35	Reserved.
34	<b>LogDeferredInMcaStat.</b> Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::L3::MCA_STATUS_L3 and MCA::L3::MCA_ADDR_L3 in addition to MCA::L3::MCA_DESTAT_L3 and MCA::L3::MCA_DEADDR_L3. 0=Only log deferred errors in MCA::L3::MCA_DESTAT_L3 and MCA::L3::MCA_DEADDR_L3. This bit does not affect logging of deferred errors in MCA::L3::MCA_SYND_L3, MCA::L3::MCA_MISC0_L3.
33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:11	Reserved.
10	<b>IntPresent.</b> Read-only, Volatile. Reset: 0. 1=This bank can be configured to trigger a corrected error interrupt using MCA::L3::MCA_CONFIG_L3[IntEn].
9	<b>McaFruTextInMca.</b> Read-write. Reset: 0. Init: BIOS,0. 1=FruText is reported McaSynd1/McaSynd2 registers
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::L3::MCA_CONFIG_L3[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::L3::MCA_CONFIG_L3[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::L3::MCA_CONFIG_L3[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::L3::MCA_CONFIG_L3[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::L3::MCA_DESTAT_L3 and MCA::L3::MCA_DEADDR_L3 are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::L3::MCA_MISC0_L3[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::L3::MCA_STATUS_L3[TCC] is present.

**MSRC000\_20[7...E]5 [L3 IP Identification] (MCA::L3::MCA\_IPID\_L3)**

Reset: 0007\_00B0\_0000\_0000h.

The MCA::L3::MCA\_IPID\_L3 register is used by software to determine what IP type and revision is associated with the MCA bank.

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2075

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2085

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2095

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A5

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B5

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C5

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D5

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E5

Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0007h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_20[7...E]6 [L3 Machine Check Syndrome] (MCA::L3::MCA\_SYND\_L3)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::L3::MCA\_STATUS\_L3 Thread 0

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2076

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2086

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2096

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A6

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B6

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C6

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D6

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E6

Bits	Description
63:32	<b>Syndrom.</b> Read-write, Volatile. Reset: Cold, 0000_0000h. Contains the syndrome, if any, associated with the error logged in MCA::L3::MCA_STATUS_L3. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::L3::MCA_SYND_L3[Length]. The Syndrome field is only valid when MCA::L3::MCA_SYND_L3[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority.</b> Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::L3::MCA_SYND_L3. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length.</b> Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::L3::MCA_SYND_L3[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::L3::MCA_SYND_L3. For example, a syndrome length of 9 means that MCA::L3::MCA_SYND_L3[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation.</b> Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 58 [MCA_SYND_L3].

Table 58: MCA\_SYND\_L3

Error Type	Bits	Description
ShadowTag	[17:12]	Reserved
	[11:8]	Pack
	[7:4]	Reserved
	[3:0]	Way

MultiHitShadowTag	[17:12] [11:8] [7:0]	Reserved Pack Reserved
Tag	[17:14] [13] [12] [11:8] [7:4] [3:0]	Reserved Reserved Reserved Bank. Reserved Way
MultiHitTag	[17:0]	Reserved
DataArray	[17:14] [13] [12] [11] [10:8] [7:4] [3:0]	Reserved Reserved Reserved Reserved Bank[2:0] Reserved Way
SdpParity	[17:0]	Reserved
XiVictimQueue	[17:0]	Reserved
Hwa	[17:0]	Reserved
XiWcbParityPoison	[17:0]	Reserved
DsmMce	[17:0]	Reserved



**MSRC000\_20[7...E]8 [L3 Machine Check Deferred Error Status] (MCA::L3::MCA\_DESTAT\_L3)**

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2078\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2088\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2098\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A8\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B8\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C8\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D8\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E8

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::L3::MCA_DEADDR_L3 contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::L3::MCA_SYND_L3. If MCA::L3::MCA_SYND_L3[ErrorPriority] is the same as the priority of the error in MCA::L3::MCA_STATUS_L3, then the information in MCA::L3::MCA_SYND_L3 is associated with the error in MCA::L3::MCA_DESTAT_L3.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h. MCA_DEFSTAT Register Reserved bits.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h. MCA_DEFSTAT Register Reserved bits.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::L3::MCA_ADDR_L3[ErrorAddr]. A value of 0 indicates that MCA::L3::MCA_ADDR_L3[63:0] contains a valid byte address. A value of 6 indicates that MCA::L3::MCA_ADDR_L3[63:6] contains a valid cache line address and that MCA::L3::MCA_ADDR_L3[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::L3::MCA_ADDR_L3[63:12] contain a valid 4KB memory page and that MCA::L3::MCA_ADDR_L3[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h. MCA_DEFSTAT Register Reserved bits.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC000\_20[7...E]9 [L3 Deferred Error Address] (MCA::L3::MCA\_DEADDR\_L3)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

The MCA::L3::MCA\_DEADDR\_L3 register stores the address associated with the error in MCA::L3::MCA\_DESTAT\_L3. The register is only meaningful if MCA::L3::MCA\_DESTAT\_L3[Val]=1 and MCA::L3::MCA\_DESTAT\_L3[AddrV]=1. The lowest valid bit of the address is defined by MCA::L3::MCA\_DESTAT\_L3[AddrLsb].

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_2079

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_2089

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_2099

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20A9

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20B9

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20C9

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20D9

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20E9

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::L3::MCA_DESTAT_L3. The lowest-order valid bit of the address is specified in MCA::L3::MCA_DESTAT_L3[AddrLsb].

**MSRC001\_040[7...E] [L3 Machine Check Control Mask] (MCA::L3::MCA\_CTL\_MASK\_L3)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC001\_0407

\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC001\_0408

\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC001\_0409

\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC001\_040A

\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC001\_040B

\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC001\_040C

\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC001\_040D

\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC001\_040E

Bits	Description
63:10	Reserved.
9	<b>DsmMce.</b> Read-write. Reset: 0. Machine check error initiated by DSM action
8	<b>XiWcbParityPoison.</b> Read-write. Reset: 0. Xi Wcb Parity Poison Creation Event
7	<b>Hwa.</b> Read-write. Reset: 0. L3 Hardware Assertion.
6	<b>XiVictimQueue.</b> Read-write. Reset: 0. L3 Victim Queue Data Fabric Error.
5	<b>SdpParity.</b> Read-write. Reset: 0. SDP Parity Error from XI.
4	<b>DataArray.</b> Read-write. Reset: 0. L3M Data ECC Error.
3	<b>MultiHitTag.</b> Read-write. Reset: 0. L3M Tag Multi-way-hit Error.
2	<b>Tag.</b> Read-write. Reset: 0. L3M Tag ECC Error.
1	<b>MultiHitShadowTag.</b> Read-write. Reset: 0. Shadow Tag Macro Multi-way-hit Error.
0	<b>ShadowTag.</b> Read-write. Reset: 0. Shadow Tag Macro ECC Error.

**MSRC000\_20[7...E]E [L3 Machine Check Syndrome Extended] (MCA::L3::MCA\_SYND1\_L3)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::L3::MCA\_STATUS\_L3 Thread 0

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_207E\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_208E\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_209E\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20AE\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20BE\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20CE\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20DE\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20EE

Bits	Description
63:0	<b>Syndrone.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::L3::MCA_SYND1_L3 register stores information associated with the error in MCA::L3::MCA_STATUS_L3 or MCA_DESTAT. The register is meaningful if MCA::L3::MCA_STATUS_L3[SyndV]=1. When MCA::L3::MCA_CONFIG_L3[McaFruTextInMca]=1, MCA::L3::MCA_SYND1_L3 stores ASCII FruText associated with the error.

**MSRC000\_20[7...E]F [L3 Machine Check Syndrome Extended] (MCA::L3::MCA\_SYND2\_L3)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::L3::MCA\_STATUS\_L3 Thread 0

\_ccd[1:0]\_lthree0\_inst7\_n[8,0]\_aliasMSR; MSRC000\_207F\_ccd[1:0]\_lthree0\_inst8\_n[9,1]\_aliasMSR; MSRC000\_208F\_ccd[1:0]\_lthree0\_inst9\_n[10,2]\_aliasMSR; MSRC000\_209F\_ccd[1:0]\_lthree0\_inst10\_n[11,3]\_aliasMSR; MSRC000\_20AF\_ccd[1:0]\_lthree0\_inst11\_n[12,4]\_aliasMSR; MSRC000\_20BF\_ccd[1:0]\_lthree0\_inst12\_n[13,5]\_aliasMSR; MSRC000\_20CF\_ccd[1:0]\_lthree0\_inst13\_n[14,6]\_aliasMSR; MSRC000\_20DF\_ccd[1:0]\_lthree0\_inst14\_n[15,7]\_aliasMSR; MSRC000\_20EF

Bits	Description
63:0	<b>Syndrone.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::L3::MCA_SYND2_L3 register stores information associated with the error in MCA::L3::MCA_STATUS_L3 or MCA_DESTAT. The register is meaningful if MCA::L3::MCA_STATUS_L3[SyndV]=1. When MCA::L3::MCA_CONFIG_L3[McaFruTextInMca]=1, MCA::L3::MCA_SYND2_L3 stores ASCII FruText associated with the error.

## 3.2.6.8 CS

<b>MSR0000_045C...MSRC000_2180 [CS Machine Check Control] (MCA::CS::MCA_CTL_CS)</b>	
Read-write. Reset: 0000_0000_0000_0000h.	
0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::CS::MCA_CTL_CS register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG_CTL. Does not affect error detection, correction, or logging.	
<u>_instCS0_n0_aliasMSRLEGACY; MSR0000_045C</u>	
<u>_instCS1_n1_aliasMSRLEGACY; MSR0000_0460</u>	
<u>_instCS0_n0_aliasMSR; MSRC000_2170</u>	
<u>_instCS1_n1_aliasMSR; MSRC000_2180</u>	
Bits	Description
63:18	Reserved.
17	<b>HWA.</b> Read-write. Reset: 0. Hardware Assert Error.
16	<b>FTI_ND_SEC_VIOL.</b> Read-write. Reset: 0. Security Violation: A security violation was detected on an incoming request from the transport layer.
15	<b>FTI_ND_ADDR_VIOL.</b> Read-write. Reset: 0. Address Violation: An address violation was detected on an incoming request from the transport layer.
14	<b>FTI_ND_ILL_REQ.</b> Read-write. Reset: 0. Illegal Request: An illegal request was received from the transport layer.
13	<b>SPF_ECC_ERR.</b> Read-write. Reset: 0. Probe Filter ECC Error: An ECC error occurred on a probe filter access.
12	<b>SPF_PRT_ERR.</b> Read-write. Reset: 0. Probe Filter Protocol Error: Indicates a Cache Coherence Issue.
11	<b>CNTR_UNFL.</b> Read-write. Reset: 0. Counter underflow error.
10	<b>CNTR_OVFL.</b> Read-write. Reset: 0. Counter overflow error.
9	<b>SDP_UNEXP_RETRY.</b> Read-write. Reset: 0. SDP read response had an unexpected RETRY error.
8	<b>SDP_RSP_NO_MTCH.</b> Read-write. Reset: 0. SDP read response had no match in the CS queue.
7	<b>ATM_PAR_ERR.</b> Read-write. Reset: 0. Atomic Request Parity Error: Parity error on read of an atomic transaction.
6	<b>SDP_PAR_ERR.</b> Read-write. Reset: 0. Read Response Parity Error: Parity error on incoming read response data.
5	<b>FTI_PAR_ERR.</b> Read-write. Reset: 0. Request or Probe Parity Error: Parity error on incoming request or probe response data.
4	<b>FTI_RSP_NO_MTCH.</b> Read-write. Reset: 0. Unexpected Response: A response was received from the transport layer which does not match any request.
3	<b>FTI_ILL_RSP.</b> Read-write. Reset: 0. Illegal Response: An illegal response was received from the transport layer.
2	<b>FTI_SEC_VIOL.</b> Read-write. Reset: 0. Security Violation: A security violation was detected on an incoming request from the transport layer.
1	<b>FTI_ADDR_VIOL.</b> Read-write. Reset: 0. Address Violation: An address violation was detected on an incoming request from the transport layer.
0	<b>FTI_ILL_REQ.</b> Read-write. Reset: 0. Illegal Request: An illegal request was received from the transport layer.

**MSR0000\_045D...MSRC000\_2181 [CS Machine Check Status] (MCA::CS::MCA\_STATUS\_CS)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_instCS0\_n0\_aliasMSRLEGACY; MSR0000\_045D

\_instCS1\_n1\_aliasMSRLEGACY; MSR0000\_0461

\_instCS0\_n0\_aliasMSR; MSRC000\_2171

\_instCS1\_n1\_aliasMSR; MSRC000\_2181

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::CS::MCA_CTL_CS. This bit is a copy of bit in MCA::CS::MCA_CTL_CS for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::CS::MCA_MISC0_CS. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::CS::MCA_ADDR_CS contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::CS::MCA_STATUS_CS[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::CS::MCA_SYND_CS. If MCA::CS::MCA_SYND_CS[ErrorPriority] is the same as the priority of the error in MCA::CS::MCA_STATUS_CS, then the information in MCA::CS::MCA_SYND_CS is associated with the error in MCA::CS::MCA_STATUS_CS. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor.

	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::CS::MCA_ADDR_CS[ErrorAddr]. A value of 0 indicates that MCA::CS::MCA_ADDR_CS[55:0] contains a valid byte address. A value of 6 indicates that MCA::CS::MCA_ADDR_CS[55:6] contains a valid cache line address and that MCA::CS::MCA_ADDR_CS[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::CS::MCA_ADDR_CS[55:12] contain a valid 4KB memory page and that MCA::CS::MCA_ADDR_CS[11:0] should be ignored by error handling software.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::CS::MCA_CTL_CS enables error reporting for the logged error.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 59: MCA\_STATUS\_CS

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
FTI_ILL_REQ	0x0	0	0	0	1	0	1
FTI_ADDR_VIOL	0x1	0	0	0	1	0	1
FTI_SEC_VIOL	0x2	0	0	0	1	0	1
FTI_ILL_RSP	0x3	1	1	1	0	0	0
FTI_RSP_NO_MTCH	0x4	1	1	1	0	0	0
FTI_PAR_ERR	0x5	0	0	0	1	0	1



SDP_PAR_ERR	0x6	0	0	0	1	0	1
ATM_PAR_ERR	0x7	0	0	0	1	0	1
SDP_RSP_NO_MTCH	0x8	1	1	1	0	0	0
SDP_UNEXP_RETRY	0x9	1	1	1	0	0	1
CNTR_OVF_L	0xa	1	1	1	0	0	0
CNTR_UNF_L	0xb	1	1	1	0	0	0
SPF_PRT_ERR	0xc	1	1	1	0	0	0
SPF_ECC_ERR	0xd	0/1	0/1	0/1	0	0	1
FTI_IL_L_REQ	0xe	0	0	0	1	0	1
FTI_ADDR_VIOL	0xf	0	0	0	1	0	1
FTI_SEC_VIOL	0x10	0	0	0	1	0	1
HWA	0x11	1	1	1	0	0	0

**MSR0000\_045E...MSRC000\_2182 [CS Machine Check Address] (MCA::CS::MCA\_ADDR\_CS)**

Reset: Cold,0000\_0000\_0000\_0000h.

MCA::CS::MCA\_ADDR\_CS stores an address and other information associated with the error in MCA::CS::MCA\_STATUS\_CS. The register is only meaningful if MCA::CS::MCA\_STATUS\_CS[Val]=1 and MCA::CS::MCA\_STATUS\_CS[AddrV]=1.

\_instCS0\_n0\_aliasMSRLEGACY; MSR0000\_045E

\_instCS1\_n1\_aliasMSRLEGACY; MSR0000\_0462

\_instCS0\_n0\_aliasMSR; MSRC000\_2172

\_instCS1\_n1\_aliasMSR; MSRC000\_2182

Bits	Description
63:56	Reserved.
55:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold,00_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::CS::MCA_STATUS_CS. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 60: MCA\_ADDR\_CS

Error Type	Bits	Description
FTI_IL_L_REQ	[47:2]	Address
FTI_ADDR_VIOL	[47:2]	Address
FTI_SEC_VIOL	[47:2]	Address
FTI_IL_L_RSP	[56:0]	Reserved
FTI_RSP_NO_MTCH	[56:0]	Reserved
FTI_PAR_ERR	[47:2]	Address
SDP_PAR_ERR	[47:2]	Address
ATM_PAR_ERR	[47:2]	Address
SDP_RSP_NO_MTCH	[56:0]	Reserved
SDP_UNEXP_RETRY	[47:2]	Address

CNTR_OVFL	[56:0]	Reserved
CNTR_UNFL	[56:0]	Reserved
SPF_PRT_ERR	[56:0]	Reserved
SPF_ECC_ERR	[47:2]	Address
FTI_ND_ILL_REQ	[47:2]	Address
FTI_ND_ADDR_VIOL	[47:2]	Address
FTI_ND_SEC_VIOL	[47:2]	Address
HWA	[47:2]	Address



**MSR0000\_045F...MSRC000\_2183 [CS Machine Check Miscellaneous 0] (MCA::CS::MCA\_MISC0\_CS)**

Log miscellaneous information associated with errors.

\_instCS0\_n0\_aliasMSRLEGACY; MSR0000\_045F

\_instCS1\_n1\_aliasMSRLEGACY; MSR0000\_0463

\_instCS0\_n0\_aliasMSR; MSRC000\_2173

\_instCS1\_n1\_aliasMSR; MSRC000\_2183

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msrr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_21[7...8]4 [CS Machine Check Configuration] (MCA::CS::MCA\_CONFIG\_CS)**

Reset: 0000\_0000\_0000\_0125h.

Controls configuration of the associated machine check bank.

\_instCS0\_n0\_aliasMSR; MSRC000\_2174

\_instCS1\_n1\_aliasMSR; MSRC000\_2184

Bits	Description
63:39	Reserved.
38:37	<b>DeferredIntType</b> . Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:35	Reserved.
34	<b>LogDeferredInMcaStat</b> . Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::CS::MCA_STATUS_CS and MCA::CS::MCA_ADDR_CS in addition to MCA::CS::MCA_DESTAT_CS and MCA::CS::MCA_DEADDR_CS. 0=Only log deferred errors in MCA::CS::MCA_DESTAT_CS and MCA::CS::MCA_DEADDR_CS. This bit does not affect logging of deferred errors in MCA::CS::MCA_SYND_CS, MCA::CS::MCA_MISC0_CS.
33	Reserved.
32	<b>McaXEnable</b> . Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:9	Reserved.
8	<b>McaLsbInStatusSupported</b> . Read-only. Reset: 1. 1=MCA::CS::MCA_CONFIG_CS[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported</b> . Read-only. Reset: 1. 1=MCA::CS::MCA_CONFIG_CS[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::CS::MCA_CONFIG_CS[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported</b> . Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::CS::MCA_CONFIG_CS[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::CS::MCA_DESTAT_CS and MCA::CS::MCA_DEADDR_CS are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX</b> . Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::CS::MCA_MISC0_CS[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::CS::MCA_STATUS_CS[TCC] is present.

**MSRC000\_21[7...8]5 [CS IP Identification] (MCA::CS::MCA\_IPID\_CS)**

Reset: 0002_002E_0000_0000h.	
The MCA::CS::MCA_IPID_CS register is used by software to determine what IP type and revision is associated with the MCA bank.	
<u>_instCS0_n0_aliasMSR; MSRC000_2175</u>	
<u>_instCS1_n1_aliasMSR; MSRC000_2185</u>	
Bits	Description
63:48	<b>McaType</b> . Read-only. Reset: 0002h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi</b> . Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID</b> . Read-only. Reset: 02Eh. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId</b> . Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_21[7...8]6 [CS Machine Check Syndrome] (MCA::CS::MCA\_SYND\_CS)**

Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h.	
Logs physical location information associated with error in MCA::CS::MCA_STATUS_CS Thread 0	
<u>_instCS0_n0_aliasMSR; MSRC000_2176</u>	
<u>_instCS1_n1_aliasMSR; MSRC000_2186</u>	
Bits	Description
63:41	Reserved.
40:32	<b>Syndrom</b> . Read-write, Volatile. Reset: Cold, 000h. Contains the syndrome, if any, associated with the error logged in MCA::CS::MCA_STATUS_CS. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::CS::MCA_SYND_CS[Length]. The Syndrome field is only valid when MCA::CS::MCA_SYND_CS[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority</b> . Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::CS::MCA_SYND_CS. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length</b> . Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::CS::MCA_SYND_CS[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::CS::MCA_SYND_CS. For example, a syndrome length of 9 means that MCA::CS::MCA_SYND_CS[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation</b> . Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 61 [MCA_SYND_CS].

Table 61: MCA\_SYND\_CS

Error Type	Bits	Description
FTI_ILL_REQ	[17:0]	Reserved
FTI_ADDR_VIOL	[17:0]	Reserved
FTI_SEC_VIOL	[17:0]	Reserved
FTI_ILL_RSP	[17:0]	Reserved
FTI_RSP_NO_MTCH	[17:0]	Reserved
FTI_PAR_ERR	[6:0]	Reserved
SDP_PAR_ERR	[6:0]	Reserved
ATM_PAR_ERR	[6:0]	Reserved
SDP_RSP_NO_MTCH	[7:0]	Reserved
SDP_UNEXP_RETRY	[6:0]	Reserved
CNTR_OVFL	[17:0]	Reserved
CNTR_UNFL	[17:0]	Reserved

SPF_PRT_ERR	[17:0]	Reserved
SPF_ECC_ERR	[17:0]	Reserved
FTI_ND_ILL_REQ	[17:0]	Reserved
FTI_ND_ADDR_VIOL	[17:0]	Reserved
FTI_ND_SEC_VIOL	[17:0]	Reserved
HWA	[17:0]	Reserved

**MSRC000\_21[7...8]8 [CS Machine Check Deferred Error Status] (MCA::CS::MCA\_DESTAT\_CS)**

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_instCS0\_n0\_aliasMSR; MSRC000\_2178

\_instCS1\_n1\_aliasMSR; MSRC000\_2188

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::CS::MCA_DEADDR_CS contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::CS::MCA_SYND_CS. If MCA::CS::MCA_SYND_CS[ErrorPriority] is the same as the priority of the error in MCA::CS::MCA_STATUS_CS, then the information in MCA::CS::MCA_SYND_CS is associated with the error in MCA::CS::MCA_DESTAT_CS.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::CS::MCA_ADDR_CS[ErrorAddr]. A value of 0 indicates that MCA::CS::MCA_ADDR_CS[55:0] contains a valid byte address. A value of 6 indicates that MCA::CS::MCA_ADDR_CS[55:6] contains a valid cache line address and that MCA::CS::MCA_ADDR_CS[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::CS::MCA_ADDR_CS[55:12] contain a valid 4KB memory page and that MCA::CS::MCA_ADDR_CS[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC000\_21[7...8]9 [CS Deferred Error Address] (MCA::CS::MCA\_DEADDR\_CS)**

Reset: Cold,0000\_0000\_0000\_0000h.

The MCA::CS::MCA\_DEADDR\_CS register stores the address associated with the error in MCA::CS::MCA\_DESTAT\_CS. The register is only meaningful if MCA::CS::MCA\_DESTAT\_CS[Val]=1 and MCA::CS::MCA\_DESTAT\_CS[AddrV]=1. The lowest valid bit of the address is defined by MCA::CS::MCA\_DESTAT\_CS[AddrLsb].

\_instCS0\_n0\_aliasMSR; MSRC000\_2179

\_instCS1\_n1\_aliasMSR; MSRC000\_2189

Bits	Description
63:56	Reserved.
55:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold,00_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::CS::MCA_DESTAT_CS. The lowest-order valid bit of the address is specified in MCA::CS::MCA_DESTAT_CS[AddrLsb].

**MSRC001\_041[7...8] [CS Machine Check Control Mask] (MCA::CS::MCA\_CTL\_MASK\_CS)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_instCS0\_n0\_aliasMSR; MSRC001\_0417

\_instCS1\_n1\_aliasMSR; MSRC001\_0418

Bits	Description
63:18	Reserved.
17	<b>HWA.</b> Read-write. Reset: 0. Hardware Assert Error.
16	<b>FTI_ND_SEC_VIOL.</b> Read-write. Reset: 0. Security Violation: A security violation was detected on an incoming request from the transport layer.
15	<b>FTI_ND_ADDR_VIOL.</b> Read-write. Reset: 0. Address Violation: An address violation was detected on an incoming request from the transport layer.
14	<b>FTI_ND_ILL_REQ.</b> Read-write. Reset: 0. Illegal Request: An illegal request was received from the transport layer.
13	<b>SPF_ECC_ERR.</b> Read-write. Reset: 0. Probe Filter ECC Error: An ECC error occurred on a probe filter access.
12	<b>SPF_PRT_ERR.</b> Read-write. Reset: 0. Probe Filter Protocol Error: Indicates a Cache Coherence Issue.
11	<b>CNTR_UNFL.</b> Read-write. Reset: 0. Counter underflow error.
10	<b>CNTR_OVFL.</b> Read-write. Reset: 0. Counter overflow error.
9	<b>SDP_UNEXP_RETRY.</b> Read-write. Reset: 0. SDP read response had an unexpected RETRY error.
8	<b>SDP_RSP_NO_MTCH.</b> Read-write. Reset: 0. SDP read response had no match in the CS queue.
7	<b>ATM_PAR_ERR.</b> Read-write. Reset: 0. Atomic Request Parity Error: Parity error on read of an atomic transaction.
6	<b>SDP_PAR_ERR.</b> Read-write. Reset: 0. Read Response Parity Error: Parity error on incoming read response data.
5	<b>FTI_PAR_ERR.</b> Read-write. Reset: 0. Request or Probe Parity Error: Parity error on incoming request or probe response data.
4	<b>FTI_RSP_NO_MTCH.</b> Read-write. Reset: 0. Unexpected Response: A response was received from the transport layer which does not match any request.
3	<b>FTI_ILL_RSP.</b> Read-write. Reset: 0. Illegal Response: An illegal response was received from the transport layer.
2	<b>FTI_SEC_VIOL.</b> Read-write. Reset: 0. Security Violation: A security violation was detected on an incoming request from the transport layer.
1	<b>FTI_ADDR_VIOL.</b> Read-write. Reset: 0. Address Violation: An address violation was detected on an incoming request from the transport layer.
0	<b>FTI_ILL_REQ.</b> Read-write. Reset: 0. Illegal Request: An illegal request was received from the transport layer.

**3.2.6.9      PIE****MSR0000\_0478...MSRC000\_21E0 [PIE Machine Check Control] (MCA::PIE::MCA\_CTL\_PIE)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::PIE::MCA\_CTL\_PIE register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_inst[PIE0,BCST]\_n0\_aliasMSRLEGACY; MSR0000\_0478

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC000\_21E0

Bits	Description
63:8	Reserved.
7	<b>RSLVFCI</b> . Read-write. Reset: 0. Register access during DF Cstate.
6	<b>CNLI</b> . Read-write. Reset: 0. Error is detected in CXL™ Link.
5	<b>WDT</b> . Read-write. Reset: 0. Watch Dog Timer: A watch dog timer expired.
4	<b>DEF</b> . Read-write. Reset: 0. A deferred error was detected in the DF.
3	<b>FTI_DAT_STAT</b> . Read-write. Reset: 0. Poison data consumption: Poison data was written to an internal PIE register.
2	<b>GMI</b> . Read-write. Reset: 0. Link Error: An error occurred on a GMI .
1	<b>CSW</b> . Read-write. Reset: 0. Register security violation: A security violation was detected on an access to an internal PIE register.
0	<b>HW_ASSERT</b> . Read-write. Reset: 0. Hardware Assert: A hardware assert was detected.



**MSR0000\_0479...MSRC000\_21E1 [PIE Machine Check Status] (MCA::PIE::MCA\_STATUS\_PIE)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_inst[PIE0,BCST]\_n0\_aliasMSRLEGACY; MSR0000\_0479

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC000\_21E1

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::PIE::MCA_CTL_PIE. This bit is a copy of bit in MCA::PIE::MCA_CTL_PIE for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::PIE::MCA_MISC0_PIE. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::PIE::MCA_ADDR_PIE contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::PIE::MCA_STATUS_PIE[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::PIE::MCA_SYND_PIE. If MCA::PIE::MCA_SYND_PIE[ErrorPriority] is the same as the priority of the error in MCA::PIE::MCA_STATUS_PIE, then the information in MCA::PIE::MCA_SYND_PIE is associated with the error in MCA::PIE::MCA_STATUS_PIE. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::PIE::MCA_ADDR_PIE[ErrorAddr]. A value of 0 indicates that MCA::PIE::MCA_ADDR_PIE[55:0] contains a valid byte address. A value of 6 indicates that MCA::PIE::MCA_ADDR_PIE[55:6] contains a valid cache line address and that MCA::PIE::MCA_ADDR_PIE[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::PIE::MCA_ADDR_PIE[55:12] contain a valid 4KB memory page and that MCA::PIE::MCA_ADDR_PIE[11:0] should be ignored by error handling software. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::PIE::MCA_CTL_PIE enables error reporting for the logged error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 62: MCA\_STATUS\_PIE

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
HW_ASSERT	0x0	1	1	1	0	0	0
CSW	0x1	0	0	0	1	0	0
GMI	0x2	0/1	0/1	0/1	0	0	0
FTI_DAT_STAT	0x3	1	1	1	0	0	0
DEF	0x4	0	0	0	1	0	0
WDT	0x5	1	1	1	0	0	1
CNLI	0x6	0/1	0/1	0/1	0	0	0
RSLVFCI	0x7	0	0	0	0/1	0	0



**MSR0000\_047A...MSRC000\_21E2 [PIE Machine Check Address] (MCA::PIE::MCA\_ADDR\_PIE)**

Reset: Cold,0000\_0000\_0000\_0000h.

MCA::PIE::MCA\_ADDR\_PIE stores an address and other information associated with the error in MCA::PIE::MCA\_STATUS\_PIE. The register is only meaningful if MCA::PIE::MCA\_STATUS\_PIE[Val]=1 and MCA::PIE::MCA\_STATUS\_PIE[AddrV]=1.

\_inst[PIE0,BCST]\_n0\_aliasMSRLEGACY; MSR0000\_047A

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC000\_21E2

Bits	Description
63:56	Reserved.
55:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold,00_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::PIE::MCA_STATUS_PIE. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 63: MCA\_ADDR\_PIE

Error Type	Bits	Description
HW_ASSERT	[56:0]	Reserved
CSW	[56:0]	Reserved
GMI	[56:0]	Reserved
FTI_DAT_STAT	[56:0]	Reserved
DEF	[56:0]	Reserved
WDT	[56:0]	Reserved
CNLI	[56:0]	Reserved
RSLVFCI	[56:0]	Reserved

**MSR0000\_047B...MSRC000\_21E3 [PIE Machine Check Miscellaneous 0] (MCA::PIE::MCA\_MISC0\_PIE)**

Log miscellaneous information associated with errors.

\_inst[PIE0,BCST]\_n0\_aliasMSRLEGACY; MSR0000\_047B

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC000\_21E3

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::PIE::MCA_MISC0_PIE[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::PIE::MCA_MISC0_PIE[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::PIE::MCA_MISC0_PIE[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::PIE::MCA_MISC0_PIE[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::PIE::MCA_MISC0_PIE[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::PIE::MCA_MISC0_PIE[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_21E4 [PIE Machine Check Configuration] (MCA::PIE::MCA\_CONFIG\_PIE)**

Reset: 0000\_0002\_0000\_0125h.

Controls configuration of the associated machine check bank.

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC000\_21E4

Bits	Description
63:39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:35	Reserved.
34	<b>LogDeferredInMcaStat.</b> Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::PIE::MCA_STATUS_PIE and MCA::PIE::MCA_ADDR_PIE in addition to MCA::PIE::MCA_DESTAT_PIE and MCA::PIE::MCA_DEADDR_PIE. 0=Only log deferred errors in MCA::PIE::MCA_DESTAT_PIE and MCA::PIE::MCA_DEADDR_PIE. This bit does not affect logging of deferred errors in MCA::PIE::MCA_SYND_PIE, MCA::PIE::MCA_MISC0_PIE.
33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:9	Reserved.
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::PIE::MCA_CONFIG_PIE[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::PIE::MCA_CONFIG_PIE[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::PIE::MCA_CONFIG_PIE[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::PIE::MCA_CONFIG_PIE[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::PIE::MCA_DESTAT_PIE and MCA::PIE::MCA_DEADDR_PIE are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::PIE::MCA_MISC0_PIE[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::PIE::MCA_STATUS_PIE[TCC] is present.

**MSRC000\_21E5 [PIE IP Identification] (MCA::PIE::MCA\_IPID\_PIE)**

Reset: 0001\_002E\_0000\_0000h.

The MCA::PIE::MCA\_IPID\_PIE register is used by software to determine what IP type and revision is associated with the MCA bank.

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC000\_21E5

Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0001h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 02Eh. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_21E6 [PIE Machine Check Syndrome] (MCA::PIE::MCA\_SYND\_PIE)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::PIE::MCA\_STATUS\_PIE Thread 0

\_inst[PIE0, BCST]\_n0\_aliasMSR; MSRC000\_21E6

Bits	Description
63:46	Reserved.
45:32	<b>Syndrom</b> . Read-write, Volatile. Reset: Cold, 0000h. Contains the syndrome, if any, associated with the error logged in MCA::PIE::MCA_STATUS_PIE. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::PIE::MCA_SYND_PIE[Length]. The Syndrome field is only valid when MCA::PIE::MCA_SYND_PIE[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority</b> . Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::PIE::MCA_SYND_PIE. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length</b> . Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::PIE::MCA_SYND_PIE[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::PIE::MCA_SYND_PIE. For example, a syndrome length of 9 means that MCA::PIE::MCA_SYND_PIE[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation</b> . Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 64 [MCA_SYND_PIE].

Table 64: MCA\_SYND\_PIE

Error Type	Bits	Description
HW_ASSERT	[17:16]	Reserved
	[15:8]	Block Instance ID
	[7:6]	Reserved
	[5:0]	Hardware Assert ID
CSW	[17]	Reserved
	[16:13]	Reserved
	[12:0]	Reserved
GMI	[17:17]	Reserved
	[16:2]	Reserved
	[1:0]	Reserved
FTI_DAT_STAT	[3:0]	Reserved
DEF	[17:0]	Reserved
WDT	[17:5]	Reserved
	[3:1]	Reserved
	[0:0]	Reserved
CNLI	[17:8]	Reserved
	[7:6]	Reserved
	[5:4]	Reserved
	[3:3]	Reserved
	[2:0]	Reserved
RSLVFCI	[17]	Reserved
	[16]	Reserved
	[15]	Reserved
	[14]	Reserved
	[13]	Reserved
	[12:0]	Reserved

**MSRC000\_21E8 [PIE Machine Check Deferred Error Status] (MCA::PIE::MCA\_DESTAT\_PIE)**

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC000\_21E8

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::PIE::MCA_DEADDR_PIE contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::PIE::MCA_SYND_PIE. If MCA::PIE::MCA_SYND_PIE[ErrorPriority] is the same as the priority of the error in MCA::PIE::MCA_STATUS_PIE, then the information in MCA::PIE::MCA_SYND_PIE is associated with the error in MCA::PIE::MCA_DESTAT_PIE.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::PIE::MCA_ADDR_PIE[ErrorAddr]. A value of 0 indicates that MCA::PIE::MCA_ADDR_PIE[55:0] contains a valid byte address. A value of 6 indicates that MCA::PIE::MCA_ADDR_PIE[55:6] contains a valid cache line address and that MCA::PIE::MCA_ADDR_PIE[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::PIE::MCA_ADDR_PIE[55:12] contain a valid 4KB memory page and that MCA::PIE::MCA_ADDR_PIE[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC000\_21E9 [PIE Deferred Error Address] (MCA::PIE::MCA\_DEADDR\_PIE)**

Reset: Cold,0000\_0000\_0000\_0000h.

The MCA::PIE::MCA\_DEADDR\_PIE register stores the address associated with the error in MCA::PIE::MCA\_DESTAT\_PIE. The register is only meaningful if MCA::PIE::MCA\_DESTAT\_PIE[Val]=1 and MCA::PIE::MCA\_DESTAT\_PIE[AddrV]=1. The lowest valid bit of the address is defined by MCA::PIE::MCA\_DESTAT\_PIE[AddrLsb].

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC000\_21E9

Bits	Description
63:56	Reserved.
55:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold,00_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::PIE::MCA_DESTAT_PIE. The lowest-order valid bit of the address is specified in MCA::PIE::MCA_DESTAT_PIE[AddrLsb].

**MSRC001\_041E [PIE Machine Check Control Mask] (MCA::PIE::MCA\_CTL\_MASK\_PIE)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_inst[PIE0,BCST]\_n0\_aliasMSR; MSRC001\_041E

Bits	Description
63:8	Reserved.
7	<b>RSLVFCI</b> . Read-write. Reset: 0. Register access during DF Cstate.
6	<b>CNLI</b> . Read-write. Reset: 0. Error is detected in CXL Link.
5	<b>WDT</b> . Read-write. Reset: 0. Watch Dog Timer: A watch dog timer expired.
4	<b>DEF</b> . Read-write. Reset: 0. A deferred error was detected in the DF.
3	<b>FTI_DAT_STAT</b> . Read-write. Reset: 0. Poison data consumption: Poison data was written to an internal PIE register.
2	<b>GMI</b> . Read-write. Reset: 0. Link Error: An error occurred on a GMI .
1	<b>CSW</b> . Read-write. Reset: 0. Register security violation: A security violation was detected on an access to an internal PIE register.
0	<b>HW_ASSERT</b> . Read-write. Reset: 0. Hardware Assert: A hardware assert was detected.

**3.2.6.10 UMC****MSR0000\_0454...MSRC000\_2160 [UMC Machine Check Control] (MCA::UMC::MCA\_CTL\_UMC)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::UMC::MCA\_CTL\_UMC register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSRLEGACY; MSR0000\_0454\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSRLEGACY; MSR0000\_0458\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2150\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2160

Bits	Description
63:12	Reserved.
11	<b>RdCrcErr</b> . Read-write. Reset: 0. Read CRC error. CRC error occurred on a DRAM read from any subchannel
10	<b>ThrttlErr</b> . Read-write. Reset: 0. Indicates that UMC is throttling
9	<b>EcsErr</b> . Read-write. Reset: 0. ECS Error. Indicates that a device exceeded the ECS Error Threshold Count.
8	<b>EcsRowErr</b> . Read-write. Reset: 0. ECS Row Error. Indicates that a single device row exceeded four code word errors.
7	<b>AesSramEccErr</b> . Read-write. Reset: 0. AES SRAM ECC error. An ECC error occurred on a AES SRAM in the processor.
6	<b>DcqSramEccErr</b> . Read-write. Reset: 0. DCQ SRAM ECC error. An ECC error occurred on a DCQ SRAM in the processor.
5	<b>WriteDataCrcErr</b> . Read-write. Reset: 0. Write data CRC error. A write data CRC error occurred on the DRAM data bus.
4	<b>AddressCommandParityErr</b> . Read-write. Reset: 0. Address/Command parity error. A parity error occurred on the DRAM address/command bus.
3	<b>ApbErr</b> . Read-write. Reset: 0. Advanced peripheral bus error. An error occurred on the advanced peripheral bus.
2	<b>SdpParityErr</b> . Read-write. Reset: 0. SDP parity error. A parity error was detected on write data from the data fabric in the processor.
1	<b>WriteDataPoisonErr</b> . Read-write. Reset: 0. Data poison error. The system tried to write poison data to DRAM and either DRAM does not support ECC or UMC_CH.EccCtrl.WrEccEn is cleared.
0	<b>DramEccErr</b> . Read-write. Reset: 0. DRAM ECC error. An ECC error occurred on a DRAM read.



**MSR0000\_0455...MSRC000\_2161 [UMC Machine Check Status] (MCA::UMC::MCA\_STATUS\_UMC)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSRLEGACY; MSR0000\_0455

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSRLEGACY; MSR0000\_0459

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2151

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2161

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::UMC::MCA_CTL_UMC. This bit is a copy of bit in MCA::UMC::MCA_CTL_UMC for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::UMC::MCA_MISC0_UMC. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::UMC::MCA_ADDR_UMC contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::UMC::MCA_STATUS_UMC[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::UMC::MCA_SYND_UMC. If MCA::UMC::MCA_SYND_UMC[ErrorPriority] is the same as the priority of the error in MCA::UMC::MCA_STATUS_UMC, then the information in MCA::UMC::MCA_SYND_UMC is associated with the error in MCA::UMC::MCA_STATUS_UMC. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::UMC::MCA_ADDR_UMC[ErrorAddr]. A value of 0 indicates that MCA::UMC::MCA_ADDR_UMC[55:0] contains a valid byte address. A value of 6 indicates that MCA::UMC::MCA_ADDR_UMC[55:6] contains a valid cache line address and that MCA::UMC::MCA_ADDR_UMC[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::UMC::MCA_ADDR_UMC[55:12] contain a valid 4KB memory page and that MCA::UMC::MCA_ADDR_UMC[11:0] should be ignored by error handling software. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::UMC::MCA_CTL_UMC enables error reporting for the logged error. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 65: MCA\_STATUS\_UMC

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
DramEccErr	0x0	0/1	0/1	0/1	0/1	0	1
WriteDataPoisonErr	0x1	1	1	1	0	0	0
SdpParityErr	0x2	1	1	1	0	0	0
ApbErr	0x3	1	1	1	0	0	1
AddressCommandParityErr	0x4	0/1	0/1	0/1	0	0	0/1



WriteDataCr cErr	0x5	0/1	0/1	0/1	0	0	0/1
DcqSramEcc Err	0x6	0/1	0/1	0/1	0	0	0
AesSramEcc Err	0x7	0/1	0/1	0/1	0	0	0
EcsRowErr	0x8	0	0	0	0	0	0
EcsErr	0x9	0	0	0	0	0	0
ThrttlErr	0xa	0	0	0	0	0	0
RdCrcErr	0xb	0	0	0	1	0	1

**MSR0000\_0456...MSRC000\_2162 [UMC Machine Check Address] (MCA::UMC::MCA\_ADDR\_UMC)**

Reset: Cold,0000\_0000\_0000\_0000h.

MCA::UMC::MCA\_ADDR\_UMC stores an address and other information associated with the error in MCA::UMC::MCA\_STATUS\_UMC. The register is only meaningful if MCA::UMC::MCA\_STATUS\_UMC[Val]=1 and MCA::UMC::MCA\_STATUS\_UMC[AddrV]=1.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSRLEGACY; MSR0000\_0456

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSRLEGACY; MSR0000\_045A

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2152

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2162

Bits	Description
63:56	Reserved.
55:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold,00_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::UMC::MCA_STATUS_UMC. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 66: MCA\_ADDR\_UMC

Error Type	Bits	Description
DramEccErr	[55:39] [39:4]	Reserved Normalized address
WriteDataPoisonErr	[56:0]	Reserved
SdpParityErr	[56:0]	Reserved
ApbErr	[55:30] [29:0]	Reserved Reserved
AddressCommandParityErr	[55:38] [37:36] [35:32] [31:0]	Reserved Reserved Chip Select Reserved
WriteDataCrcErr	[55:38] [37:36] [35:32] [31:0]	Reserved Reserved Chip Select Reserved
DcqSramEccErr	[56:0]	Reserved
AesSramEccErr	[56:0]	Reserved
EcsRowErr	[56:0]	Reserved
EcsErr	[56:0]	Reserved
ThrttlErr	[56:0]	Reserved
RdCrcErr	[55:39] [39:4]	Reserved Normalized address

**MSR0000\_0457...MSRC000\_2163 [UMC Machine Check Miscellaneous 0] (MCA::UMC::MCA\_MISC0\_UMC)**

Log miscellaneous information associated with errors.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSRLEGACY; MSR0000\_0457

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSRLEGACY; MSR0000\_045B

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2153

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2163

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC0_UMC[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC0_UMC[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC0_UMC[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC0_UMC[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC0_UMC[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC0_UMC[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_21[5...6]4 [UMC Machine Check Configuration] (MCA::UMC::MCA\_CONFIG\_UMC)**

Reset: 0000\_0002\_0000\_0125h.

Controls configuration of the associated machine check bank.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2154

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2164

Bits	Description
63:39	Reserved.
38:37	<b>DeferredIntType</b> . Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:35	Reserved.
34	<b>LogDeferredInMcaStat</b> . Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::UMC::MCA_STATUS_UMC and MCA::UMC::MCA_ADDR_UMC in addition to MCA::UMC::MCA_DESTAT_UMC and MCA::UMC::MCA_DEADDR_UMC. 0=Only log deferred errors in MCA::UMC::MCA_DESTAT_UMC and MCA::UMC::MCA_DEADDR_UMC. This bit does not affect logging of deferred errors in MCA::UMC::MCA_SYND_UMC, MCA::UMC::MCA_MISC0_UMC.
33	Reserved.
32	<b>McaXEnable</b> . Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:9	Reserved.
8	<b>McaLsbInStatusSupported</b> . Read-only. Reset: 1. 1=MCA::UMC::MCA_CONFIG_UMC[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported</b> . Read-only. Reset: 1. 1=MCA::UMC::MCA_CONFIG_UMC[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::UMC::MCA_CONFIG_UMC[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported</b> . Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::UMC::MCA_CONFIG_UMC[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::UMC::MCA_DESTAT_UMC and MCA::UMC::MCA_DEADDR_UMC are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX</b> . Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::UMC::MCA_MISC0_UMC[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::UMC::MCA_STATUS_UMC[TCC] is present.

**MSRC000\_21[5...6]5 [UMC IP Identification] (MCA::UMC::MCA\_IPID\_UMC)**

Reset: 0000\_0096\_0000\_0000h.

The MCA::UMC::MCA\_IPID\_UMC register is used by software to determine what IP type and revision is associated with the MCA bank.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2155

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2165

Bits	Description
63:48	<b>McaType</b> . Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi</b> . Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID</b> . Read-only. Reset: 096h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId</b> . Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_21[5...6]6 [UMC Machine Check Syndrome] (MCA::UMC::MCA\_SYND\_UMC)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::UMC::MCA\_STATUS\_UMC Thread 0

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2156

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2166

Bits	Description
63:32	<b>Syndrom</b> . Read-write, Volatile. Reset: Cold, 0000_0000h. Contains the syndrome, if any, associated with the error logged in MCA::UMC::MCA_STATUS_UMC. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::UMC::MCA_SYND_UMC[Length]. The Syndrome field is only valid when MCA::UMC::MCA_SYND_UMC[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority</b> . Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::UMC::MCA_SYND_UMC. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length</b> . Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::UMC::MCA_SYND_UMC[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::UMC::MCA_SYND_UMC. For example, a syndrome length of 9 means that MCA::UMC::MCA_SYND_UMC[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation</b> . Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 67 [MCA_SYND_UMC].

Table 67: MCA\_SYND\_UMC

Error Type	Bits	Description
DramEccErr	[17:16] [15] [14] [13:8] [7:4] [3] [2:0]	Reserved Reserved Reserved Symbol. Only contains valid information on a corrected error. Cid. Specifies the rank multiply ID for supported DIMMs. Sub channel Chip Select
WriteDataPoisonErr	[17:0]	Reserved
SdpParityErr	[17:0]	Reserved
ApbErr	[17:0]	Reserved
AddressCommandParityErr	[17:0]	Reserved
WriteDataCrcErr	[17:0]	Reserved
DcqSramEccErr	[17:14]	Reserved

	[13:0]	Reserved
AesSramEccErr	[17]	Reserved
	[16:8]	Reserved
	[7:4]	Reserved
	[3:2]	Reserved
	[1:0]	Reserved
EcsRowErr	[17:0]	Reserved
EcsErr	[17:0]	Reserved
ThrttlErr	[17:0]	Reserved
RdCrcErr	[17:8]	Reserved
	[7:4]	Cid. Specifies the rank multiply ID for supported DIMMs.
	[3]	Sub channel
	[2:0]	Chip Select

**MSRC000\_21[5...6]E [UMC Machine Check Syndrome Extended] (MCA::UMC::MCA\_SYND1\_UMC)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::UMC::MCA\_STATUS\_UMC Thread 0

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_215E

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_216E

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::UMC::MCA_SYND1_UMC register stores information associated with the error in MCA::UMC::MCA_STATUS_UMC or MCA_DESTAT. The register is meaningful if MCA::UMC::MCA_STATUS_UMC[SyndV]=1. When MCA::UMC::MCA_CONFIG_UMC[McaFruTextInMca]=1, MCA::UMC::MCA_SYND1_UMC stores ASCII FruText associated with the error.

**MSRC000\_21[5...6]F [UMC Machine Check Syndrome Extended] (MCA::UMC::MCA\_SYND2\_UMC)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::UMC::MCA\_STATUS\_UMC Thread 0

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_215F

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_216F

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::UMC::MCA_SYND2_UMC register stores information associated with the error in MCA::UMC::MCA_STATUS_UMC or MCA_DESTAT. The register is meaningful if MCA::UMC::MCA_STATUS_UMC[SyndV]=1. When MCA::UMC::MCA_CONFIG_UMC[McaFruTextInMca]=1, MCA::UMC::MCA_SYND2_UMC stores ASCII FruText associated with the error.

**MSRC000\_21[5...6]8 [UMC Machine Check Deferred Error Status] (MCA::UMC::MCA\_DESTAT\_UMC)**

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2158

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2168

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::UMC::MCA_DEADDR_UMC contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::UMC::MCA_SYND_UMC. If MCA::UMC::MCA_SYND_UMC[ErrorPriority] is the same as the priority of the error in MCA::UMC::MCA_STATUS_UMC, then the information in MCA::UMC::MCA_SYND_UMC is associated with the error in MCA::UMC::MCA_DESTAT_UMC.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::UMC::MCA_ADDR_UMC[ErrorAddr]. A value of 0 indicates that MCA::UMC::MCA_ADDR_UMC[55:0] contains a valid byte address. A value of 6 indicates that MCA::UMC::MCA_ADDR_UMC[55:6] contains a valid cache line address and that MCA::UMC::MCA_ADDR_UMC[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::UMC::MCA_ADDR_UMC[55:12] contain a valid 4KB memory page and that MCA::UMC::MCA_ADDR_UMC[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC000\_21[5...6]9 [UMC Deferred Error Address] (MCA::UMC::MCA\_DEADDR\_UMC)**

Reset: Cold,0000\_0000\_0000\_0000h.

The MCA::UMC::MCA\_DEADDR\_UMC register stores the address associated with the error in MCA::UMC::MCA\_DESTAT\_UMC. The register is only meaningful if MCA::UMC::MCA\_DESTAT\_UMC[Val]=1 and MCA::UMC::MCA\_DESTAT\_UMC[AddrV]=1. The lowest valid bit of the address is defined by MCA::UMC::MCA\_DESTAT\_UMC[AddrLsb].

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_2159

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_2169

Bits	Description
63:56	Reserved.
55:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold,00_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::UMC::MCA_DESTAT_UMC. The lowest-order valid bit of the address is specified in MCA::UMC::MCA_DESTAT_UMC[AddrLsb].



**MSRC000\_21[5...6]A [UMC Machine Check Miscellaneous 1] (MCA::UMC::MCA\_MISC1\_UMC)**

Log miscellaneous information associated with errors, as defined by each error type.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC000\_215A

\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC000\_216A

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC1_UMC[Locked]) ? Read-write : Read-only.
59:52	Reserved.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC1_UMC[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msrr::McalntrCfg[ThresholdLvtOffset]) to all cores. 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC1_UMC[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh; also set by hardware if ErrCnt is initialized to FFFh and transitions from FFFh to 000h. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC1_UMC[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::UMC::MCA_MISC1_UMC[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC001\_041[5...6] [UMC Machine Check Control Mask] (MCA::UMC::MCA\_CTL\_MASK\_UMC)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_instUMCWPHY0UMC\_n0\_umc0\_aliasMSR; MSRC001\_0415\_instUMCWPHY1UMC\_n1\_umc0\_aliasMSR; MSRC001\_0416

Bits	Description
63:12	Reserved.
11	<b>RdCrcErr.</b> Read-write. Reset: 0. Read CRC error. CRC error occurred on a DRAM read from any subchannel
10	<b>ThrttlErr.</b> Read-write. Reset: 0. Indicates that UMC is throttling
9	<b>EcsErr.</b> Read-write. Reset: 0. ECS Error. Indicates that a device exceeded the ECS Error Threshold Count.
8	<b>EcsRowErr.</b> Read-write. Reset: 0. ECS Row Error. Indicates that a single device row exceeded four code word errors.
7	<b>AesSramEccErr.</b> Read-write. Reset: 0. AES SRAM ECC error. An ECC error occurred on a AES SRAM in the processor.
6	<b>DcqSramEccErr.</b> Read-write. Reset: 0. DCQ SRAM ECC error. An ECC error occurred on a DCQ SRAM in the processor.
5	<b>WriteDataCrcErr.</b> Read-write. Reset: 0. Write data CRC error. A write data CRC error occurred on the DRAM data bus.
4	<b>AddressCommandParityErr.</b> Read-write. Reset: 0. Address/Command parity error. A parity error occurred on the DRAM address/command bus.
3	<b>ApbErr.</b> Read-write. Reset: 0. Advanced peripheral bus error. An error occurred on the advanced peripheral bus.
2	<b>SdpParityErr.</b> Read-write. Reset: 0. SDP parity error. A parity error was detected on write data from the data fabric in the processor.
1	<b>WriteDataPoisonErr.</b> Read-write. Reset: 0. Data poison error. The system tried to write poison data to DRAM and either DRAM does not support ECC or UMC_CH.EccCtrl.WrEccEn is cleared.
0	<b>DramEccErr.</b> Read-write. Reset: 0. DRAM ECC error. An ECC error occurred on a DRAM read.

**3.2.6.11 MP5****MSR0000\_043C...MSRC000\_20F0 (MCA::MP5::MCA\_CTL\_MP5)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSRLEGACY; MSR0000\_043C\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC000\_20F0

Bits	Description
63:11	Reserved.
10	<b>Mp5FuseSramError.</b> Read-write. Reset: 0. Fuse SRAM ECC or parity error.
9	<b>Mp5ITagBError.</b> Read-write. Reset: 0. Instruction Tag Cache Bank B ECC or parity error.
8	<b>Mp5ITagAError.</b> Read-write. Reset: 0. Instruction Tag Cache Bank A ECC or parity error.
7	<b>Mp5ICacheBError.</b> Read-write. Reset: 0. Instruction Cache Bank B ECC or parity error.
6	<b>Mp5ICacheAError.</b> Read-write. Reset: 0. Instruction Cache Bank A ECC or parity error.
5	<b>Mp5DTagBError.</b> Read-write. Reset: 0. Data Tag Cache Bank B ECC or parity error.
4	<b>Mp5DTagAError.</b> Read-write. Reset: 0. Data Tag Cache Bank A ECC or parity error.
3	<b>Mp5DCacheBError.</b> Read-write. Reset: 0. Data Cache Bank B ECC or parity error.
2	<b>Mp5DCacheAError.</b> Read-write. Reset: 0. Data Cache Bank A ECC or parity error.
1	<b>Mp5LowSramError.</b> Read-write. Reset: 0. Low SRAM ECC or parity error.
0	<b>Mp5HighSramError.</b> Read-write. Reset: 0. High SRAM ECC or parity error.



**MSR0000\_043D...MSRC000\_20F1 [MP5 Machine Check Status] (MCA::MP5::MCA\_STATUS\_MP5)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSRLEGACY; MSR0000\_043D

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC000\_20F1

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::MP5::MCA_CTL_MP5. This bit is a copy of bit in MCA::MP5::MCA_CTL_MP5 for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::MP5::MCA_MISC0_MP5. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::MP5::MCA_ADDR_MP5 contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::MP5::MCA_STATUS_MP5[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::MP5::MCA_SYND_MP5. If MCA::MP5::MCA_SYND_MP5[ErrorPriority] is the same as the priority of the error in MCA::MP5::MCA_STATUS_MP5, then the information in MCA::MP5::MCA_SYND_MP5 is associated with the error in MCA::MP5::MCA_STATUS_MP5. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor.

	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::MP5::MCA_ADDR_MP5[ErrorAddr]. A value of 0 indicates that MCA::MP5::MCA_ADDR_MP5[63:0] contains a valid byte address. A value of 6 indicates that MCA::MP5::MCA_ADDR_MP5[63:6] contains a valid cache line address and that MCA::MP5::MCA_ADDR_MP5[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::MP5::MCA_ADDR_MP5[63:12] contain a valid 4KB memory page and that MCA::MP5::MCA_ADDR_MP5[11:0] should be ignored by error handling software.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::MP5::MCA_CTL_MP5 enables error reporting for the logged error.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 68: MCA\_STATUS\_MP5

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
Mp5HighSramError	0x0	0/1	0/1	0/1	0	0	1
Mp5LowSramError	0x1	0/1	0/1	0/1	0	0	1
Mp5DCacheAError	0x2	0/1	0/1	0/1	0	0	1
Mp5DCacheBError	0x3	0/1	0/1	0/1	0	0	1
Mp5DTagAError	0x4	0/1	0/1	0/1	0	0	1
Mp5DTagB	0x5	0/1	0/1	0/1	0	0	1

Error							
Mp5ICacheAError	0x6	0/1	0/1	0/1	0	0	1
Mp5ICacheBError	0x7	0/1	0/1	0/1	0	0	1
Mp5ITagAError	0x8	0/1	0/1	0/1	0	0	1
Mp5ITagBError	0x9	0/1	0/1	0/1	0	0	1
Mp5FuseSramError	0xa	0/1	0/1	0/1	0	0	1

**MSR0000\_043E...MSRC000\_20F2 (MCA::MP5::MCA\_ADDR\_MP5)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSRLEGACY; MSR0000\_043E

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC000\_20F2

Bits	Description
63:0	<b>ErrorAddr.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. Unless otherwise specified by an error, contains the address associated with the error logged in MCA::MP5::MCA_STATUS_MP5. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 69: MCA\_ADDR\_MP5

Error Type	Bits	Description
Mp5HighSramError	[55:0]	Reserved
Mp5LowSramError	[55:0]	Reserved
Mp5DCacheAError	[55:0]	Reserved
Mp5DCacheBError	[55:0]	Reserved
Mp5DTagAError	[55:0]	Reserved
Mp5DTagBError	[55:0]	Reserved
Mp5ICacheAError	[55:0]	Reserved
Mp5ICacheBError	[55:0]	Reserved
Mp5ITagAError	[55:0]	Reserved
Mp5ITagBError	[55:0]	Reserved
Mp5FuseSramError	[55:0]	Reserved

**MSR0000\_043F...MSRC000\_20F3 (MCA::MP5::MCA\_MISC0\_MP5)**

_ccd[1:0]_instMP5_n[1:0]_aliasMSRLEGACY; MSR0000_043F	
_ccd[1:0]_instMP5_n[1:0]_aliasMSR; MSRC000_20F3	
Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::MP5::MCA_MISC0_MP5[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::MP5::MCA_MISC0_MP5[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::MP5::MCA_MISC0_MP5[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::MP5::MCA_MISC0_MP5[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::MP5::MCA_MISC0_MP5[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::MP5::MCA_MISC0_MP5[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_20F4 [MP5 Machine Check Configuration] (MCA::MP5::MCA\_CONFIG\_MP5)**

Reset: 0000\_0000\_0000\_0121h.

Controls configuration of the associated machine check bank.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC000\_20F4

Bits	Description
63:41	Reserved.
40	<b>IntEn.</b> Read-write. Reset: 0. Init: BIOS,0. 1=When set, this bank will generate corrected error interrupts.
39	Reserved.
38:37	<b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:33	Reserved.
32	<b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:11	Reserved.
10	<b>IntPresent.</b> Read-only, Volatile. Reset: 0. 1=This bank can be configured to trigger a corrected error interrupt using MCA::MP5::MCA_CONFIG_MP5[IntEn].
9	<b>McaFruTextInMca.</b> Read-write. Reset: 0. Init: BIOS,0. 1=FruText is reported McaSynd1/McaSynd2 registers
8	<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::MP5::MCA_CONFIG_MP5[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::MP5::MCA_CONFIG_MP5[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::MP5::MCA_CONFIG_MP5[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 0. 1=Deferred errors are supported in this MCA bank, and the LogDeferredInMcaStat field in this register controls the logging behavior of these errors. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::MP5::MCA_MISC0_MP5[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::MP5::MCA_STATUS_MP5[TCC] is present.

**MSRC000\_20F5 (MCA::MP5::MCA\_IPID\_MP5)**

Reset: 0002\_0001\_0000\_0000h.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC000\_20F5

Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0002h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 001h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_20F6 (MCA::MP5::MCA\_SYND\_MP5)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC000\_20F6

Bits	Description
63:27	Reserved.
26:24	<b>ErrorPriority.</b> Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::MP5::MCA_SYND_MP5. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length.</b> Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of any syndromes logged. Only meaningful if the Syndrome field exists in this register.
17:0	<b>ErrorInformation.</b> Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 70 [MCA_SYND_MP5].

Table 70: MCA\_SYND\_MP5

Error Type	Bits	Description
Mp5HighSramError	[17:15]	Reserved
	[14:0]	Reserved
Mp5LowSramError	[17:15]	Reserved
	[14:0]	Reserved
Mp5DCacheAError	[17:8]	Reserved
	[7:0]	Reserved
Mp5DCacheBError	[17:8]	Reserved
	[7:0]	Reserved
Mp5DTagAError	[17:7]	Reserved
	[6:0]	Reserved
Mp5DTagBError	[17:7]	Reserved
	[6:0]	Reserved
Mp5ICacheAError	[17:8]	Reserved
	[7:0]	Reserved
Mp5ICacheBError	[17:8]	Reserved
	[7:0]	Reserved
Mp5ITagAError	[17:6]	Reserved
	[5:0]	Reserved
Mp5ITagBError	[17:6]	Reserved
	[5:0]	Reserved
Mp5FuseSramError	[17:15]	Reserved
	[14:0]	Reserved

**MSRC001\_040F (MCA::MP5::MCA\_CTL\_MASK\_MP5)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC001\_040F

Bits	Description
63:11	Reserved.
10	<b>Mp5FuseSramError.</b> Read-write. Reset: 0. Fuse SRAM ECC or parity error.
9	<b>Mp5ITagBError.</b> Read-write. Reset: 0. Instruction Tag Cache Bank B ECC or parity error.
8	<b>Mp5ITagAError.</b> Read-write. Reset: 0. Instruction Tag Cache Bank A ECC or parity error.
7	<b>Mp5ICacheBError.</b> Read-write. Reset: 0. Instruction Cache Bank B ECC or parity error.
6	<b>Mp5ICacheAError.</b> Read-write. Reset: 0. Instruction Cache Bank A ECC or parity error.
5	<b>Mp5DTagBError.</b> Read-write. Reset: 0. Data Tag Cache Bank B ECC or parity error.
4	<b>Mp5DTagAError.</b> Read-write. Reset: 0. Data Tag Cache Bank A ECC or parity error.
3	<b>Mp5DCacheBError.</b> Read-write. Reset: 0. Data Cache Bank B ECC or parity error.
2	<b>Mp5DCacheAError.</b> Read-write. Reset: 0. Data Cache Bank A ECC or parity error.
1	<b>Mp5LowSramError.</b> Read-write. Reset: 0. Low SRAM ECC or parity error.
0	<b>Mp5HighSramError.</b> Read-write. Reset: 0. High SRAM ECC or parity error.

**MSRC000\_20FE (MCA::MP5::MCA\_SYND1\_MP5)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC000\_20FE

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::MP5::MCA_SYND1_MP5 register stores information associated with the error in MCA::MP5::MCA_STATUS_MP5 or MCA_DESTAT. The register is meaningful if MCA::MP5::MCA_STATUS_MP5[SyndV]=1. When MCA::MP5::MCA_CONFIG_MP5[McaFruTextInMca]=1, MCA::MP5::MCA_SYND1_MP5 stores ASCII FruText associated with the error.

**MSRC000\_20FF (MCA::MP5::MCA\_SYND2\_MP5)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

\_ccd[1:0]\_instMP5\_n[1:0]\_aliasMSR; MSRC000\_20FF

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::MP5::MCA_SYND2_MP5 register stores information associated with the error in MCA::MP5::MCA_STATUS_MP5 or MCA_DESTAT. The register is meaningful if MCA::MP5::MCA_STATUS_MP5[SyndV]=1. When MCA::MP5::MCA_CONFIG_MP5[McaFruTextInMca]=1, MCA::MP5::MCA_SYND2_MP5 stores ASCII FruText associated with the error.



## 3.2.6.12 NBIO

**MSR0000\_046C...MSRC000\_21B0 [NBIO Machine Check Control] (MCA::NBIO::MCA\_CTL\_NBIO)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::NBIO::MCA\_CTL\_NBIO register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_instIOHC\_n0\_aliasMSRLEGACY; MSR0000\_046C

\_instIOHC\_n0\_aliasMSR; MSRC000\_21B0

Bits	Description
63:6	Reserved.
5	<b>Int_ErrEvent.</b> Read-write. Reset: 0. Internal system fatal error event was detected.
4	<b>IOHC_Internal_Poison.</b> Read-write. Reset: 0. Internal Poison Error. Poison data was sent to an internal client.
3	<b>Egress_Poison.</b> Read-write. Reset: 0. SDP Egress Poison Error. Poison was propagated to an egress port.
2	<b>Ext_ErrEvent.</b> Read-write. Reset: 0. External SDP ErrEvent error. A system fatal error event from an SDP interface was detected.
1	<b>PCIE_Sideband.</b> Read-write. Reset: 0. PCIE error. A PCIe® error was logged in a PCIe® root port.
0	<b>EccParityError.</b> Read-write. Reset: 0. ECC or Parity error. An SRAM ECC or parity error was detected.



**MSR0000\_046D...MSRC000\_21B1 [NBIO Machine Check Status] (MCA::NBIO::MCA\_STATUS\_NBIO)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_instIOHC\_n0\_aliasMSRLEGACY; MSR0000\_046D

\_instIOHC\_n0\_aliasMSR; MSRC000\_21B1

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::NBIO::MCA_CTL_NBIO. This bit is a copy of bit in MCA::NBIO::MCA_CTL_NBIO for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::NBIO::MCA_MISC0_NBIO. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::NBIO::MCA_ADDR_NBIO contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::NBIO::MCA_STATUS_NBIO[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::NBIO::MCA_SYND_NBIO. If MCA::NBIO::MCA_SYND_NBIO[ErrorPriority] is the same as the priority of the error in MCA::NBIO::MCA_STATUS_NBIO, then the information in MCA::NBIO::MCA_SYND_NBIO is associated with the error in MCA::NBIO::MCA_STATUS_NBIO. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC.</b> Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor.

	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred.</b> Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison.</b> Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub.</b> Reset: Cold,0. 1=The error was the result of a scrub operation.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId.</b> Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb.</b> Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::NBIO::MCA_ADDR_NBIO[ErrorAddr]. A value of 0 indicates that MCA::NBIO::MCA_ADDR_NBIO[55:0] contains a valid byte address. A value of 6 indicates that MCA::NBIO::MCA_ADDR_NBIO[55:6] contains a valid cache line address and that MCA::NBIO::MCA_ADDR_NBIO[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::NBIO::MCA_ADDR_NBIO[55:12] contain a valid 4KB memory page and that MCA::NBIO::MCA_ADDR_NBIO[11:0] should be ignored by error handling software.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22.</b> Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt.</b> Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::NBIO::MCA_CTL_NBIO enables error reporting for the logged error.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode.</b> Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 71: MCA\_STATUS\_NBIO

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
EccParityError	0x0	0/1	0/1	0/1	0/1	0	0
PCIE_Sideband	0x1	0/1	0/1	0/1	0/1	0	0
Ext_ErrEvent	0x2	1	1	1	0	0	0
Egress_Poison	0x3	0/1	0/1	0/1	0/1	0	0
IOHC_Internal_Poison	0x4	1	1	1	0	0	0
Int_ErrEvent	0x5	1	1	1	0	0	0

**MSR0000\_046E...MSRC000\_21B2 [NBIO Machine Check Address] (MCA::NBIO::MCA\_ADDR\_NBIO)**

Read-only. Reset: Cold,0000\_0000\_0000\_0000h.

MCA::NBIO::MCA\_ADDR\_NBIO stores an address and other information associated with the error in MCA::NBIO::MCA\_STATUS\_NBIO. The register is only meaningful if MCA::NBIO::MCA\_STATUS\_NBIO[Val]=1 and MCA::NBIO::MCA\_STATUS\_NBIO[AddrV]=1.

\_instIOHC\_n0\_aliasMSRLEGACY; MSR0000\_046E

\_instIOHC\_n0\_aliasMSR; MSRC000\_21B2

Bits	Description
63:0	<b>ErrorAddr.</b> Read-only. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::NBIO::MCA_STATUS_NBIO.

Table 72: MCA\_ADDR\_NBIO

Error Type	Bits	Description
EccParityError	[56:0]	Reserved
PCIE_Sideband	[56:0]	Reserved
Ext_ErrEvent	[56:0]	Reserved
Egress_Poison	[56:0]	Reserved
IOHC_Internal_Poison	[56:0]	Reserved
Int_ErrEvent	[56:0]	Reserved

**MSR0000\_046F...MSRC000\_21B3 [NBIO Machine Check Miscellaneous 0] (MCA::NBIO::MCA\_MISC0\_NBIO)**

Log miscellaneous information associated with errors.

\_instIOHC\_n0\_aliasMSRLEGACY; MSR0000\_046F

\_instIOHC\_n0\_aliasMSR; MSRC000\_21B3

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::NBIO::MCA_MISC0_NBIO[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::NBIO::MCA_MISC0_NBIO[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::NBIO::MCA_MISC0_NBIO[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::NBIO::MCA_MISC0_NBIO[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::NBIO::MCA_MISC0_NBIO[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   !MCA::NBIO::MCA_MISC0_NBIO[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_21B4 [NBIO Machine Check Configuration] (MCA::NBIO::MCA\_CONFIG\_NBIO)**

Reset: 0000\_0002\_0000\_0125h.

Controls configuration of the associated machine check bank.

\_instIOHC\_n0\_aliasMSR; MSRC000\_21B4

Bits	Description
63:39	Reserved.
38:37	<b>DeferredIntType</b> . Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:35	Reserved.
34	<b>LogDeferredInMcaStat</b> . Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::NBIO::MCA_STATUS_NBIO and MCA::NBIO::MCA_ADDR_NBIO in addition to MCA::NBIO::MCA_DESTAT_NBIO and MCA::NBIO::MCA_DEADDR_NBIO. 0=Only log deferred errors in MCA::NBIO::MCA_DESTAT_NBIO and MCA::NBIO::MCA_DEADDR_NBIO. This bit does not affect logging of deferred errors in MCA::NBIO::MCA_SYND_NBIO, MCA::NBIO::MCA_MISC0_NBIO.
33	Reserved.
32	<b>McaXEnable</b> . Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:9	Reserved.
8	<b>McaLsbInStatusSupported</b> . Read-only. Reset: 1. 1=MCA::NBIO::MCA_CONFIG_NBIO[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported</b> . Read-only. Reset: 1. 1=MCA::NBIO::MCA_CONFIG_NBIO[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::NBIO::MCA_CONFIG_NBIO[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported</b> . Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::NBIO::MCA_CONFIG_NBIO[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::NBIO::MCA_DESTAT_NBIO and MCA::NBIO::MCA_DEADDR_NBIO are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX</b> . Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::NBIO::MCA_MISC0_NBIO[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::NBIO::MCA_STATUS_NBIO[TCC] is present.

**MSRC000\_21B5 [NBIO IP Identification] (MCA::NBIO::MCA\_IPID\_NBIO)**

Reset: 0000\_0018\_0000\_0000h.

The MCA::NBIO::MCA\_IPID\_NBIO register is used by software to determine what IP type and revision is associated with the MCA bank.

\_instIOHC\_n0\_aliasMSR; MSRC000\_21B5

Bits	Description
63:48	<b>McaType</b> . Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi</b> . Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID</b> . Read-only. Reset: 018h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId</b> . Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

<b>MSRC000_21B6 [NBIO Machine Check Syndrome] (MCA::NBIO::MCA_SYND_NBIO)</b>	
Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h.	
Logs physical location information associated with error in MCA::NBIO::MCA_STATUS_NBIO Thread 0 _instIOHC_n0_aliasMSR; MSRC000_21B6	
<b>Bits</b>	<b>Description</b>
63:33	Reserved.
32	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0. Contains the syndrome, if any, associated with the error logged in MCA::NBIO::MCA_STATUS_NBIO. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::NBIO::MCA_SYND_NBIO[Length]. The Syndrome field is only valid when MCA::NBIO::MCA_SYND_NBIO[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority.</b> Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::NBIO::MCA_SYND_NBIO. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length.</b> Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::NBIO::MCA_SYND_NBIO[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::NBIO::MCA_SYND_NBIO. For example, a syndrome length of 9 means that MCA::NBIO::MCA_SYND_NBIO[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation.</b> Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 73 [MCA_SYND_NBIO].

Table 73: MCA\_SYND\_NBIO

<b>Error Type</b>	<b>Bits</b>	<b>Description</b>
EccParityError	[17:5] [4:0]	Group ID Structure ID
PCIE_Sideband	[5:0]	EgressPortNum
Ext_ErrEvent	[3:0]	Reserved
Egress_Poison	[5:0]	Egress Port Number
IOHC_Internal_Poison	[0]	0:CfgMaster 1:TrapClient
Int_ErrEvent	[0]	Reserved



**MSRC000\_21B8 [NBIO Machine Check Deferred Error Status] (MCA::NBIO::MCA\_DESTAT\_NBIO)**

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_instIOHC\_n0\_aliasMSR; MSRC000\_21B8

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::NBIO::MCA_DEADDR_NBIO contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::NBIO::MCA_SYND_NBIO. If MCA::NBIO::MCA_SYND_NBIO[ErrorPriority] is the same as the priority of the error in MCA::NBIO::MCA_STATUS_NBIO, then the information in MCA::NBIO::MCA_SYND_NBIO is associated with the error in MCA::NBIO::MCA_DESTAT_NBIO.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::NBIO::MCA_ADDR_NBIO[ErrorAddr]. A value of 0 indicates that MCA::NBIO::MCA_ADDR_NBIO[55:0] contains a valid byte address. A value of 6 indicates that MCA::NBIO::MCA_ADDR_NBIO[55:6] contains a valid cache line address and that MCA::NBIO::MCA_ADDR_NBIO[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::NBIO::MCA_ADDR_NBIO[55:12] contain a valid 4KB memory page and that MCA::NBIO::MCA_ADDR_NBIO[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC001\_041B [NBIO Machine Check Control Mask] (MCA::NBIO::MCA\_CTL\_MASK\_NBIO)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_instIOHC\_n0\_aliasMSR; MSRC001\_041B

Bits	Description
63:6	Reserved.
5	<b>Int_ErrEvent.</b> Read-write. Reset: 0. Init: BIOS,1. Internal system fatal error event was detected.
4	<b>IOHC_Internal_Poison.</b> Read-write. Reset: 0. Internal Poison Error. Poison data was sent to an internal client.
3	<b>Egress_Poison.</b> Read-write. Reset: 0. SDP Egress Poison Error. Poison was propagated to an egress port.
2	<b>Ext_ErrEvent.</b> Read-write. Reset: 0. Init: BIOS,1. External SDP ErrEvent error. A system fatal error event from an SDP interface was detected.
1	<b>PCIE_Sideband.</b> Read-write. Reset: 0. Init: BIOS,1. PCIe error. A PCIe® error was logged in a PCIe® root port.
0	<b>EccParityError.</b> Read-write. Reset: 0. ECC or Parity error. An SRAM ECC or parity error was detected.

**3.2.6.13 KPX SERDES****MSR0000\_047C...MSRC000\_21F0 [KPX\_SERDES Machine Check Control]  
(MCA::KPX::SERDES::MCA\_CTL\_KPX\_SERDES)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::KPX::SERDES::MCA\_CTL\_KPX\_SERDES register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_instKPXSERDES\_n[1:0]\_aliasMSRLEGACY; MSR0000\_047C

\_instKPXSERDES\_n[1:0]\_aliasMSR; MSRC000\_21F0

Bits	Description
63:4	Reserved.
3	<b>APB.</b> Read-write. Reset: 0. PHY APB error
2	<b>ARCData.</b> Read-write. Reset: 0. ARC data buffer parity error
1	<b>ARCIns.</b> Read-write. Reset: 0. ARC instruction buffer parity error
0	<b>RAMECC.</b> Read-write. Reset: 0. RAM ECC Error.



**MSR0000\_047D...MSRC000\_21F1 [KPX\_SERDES Machine Check Status]  
(MCA::KPX::SERDES::MCA\_STATUS\_KPX\_SERDES)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_instKPXSERDES\_n[1:0]\_aliasMSRLEGACY; MSR0000\_047D

\_instKPXSERDES\_n[1:0]\_aliasMSR; MSRC000\_21F1

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::KPX::SERDES::MCA_CTL_KPX_SERDES. This bit is a copy of bit in MCA::KPX::SERDES::MCA_CTL_KPX_SERDES for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::KPX::SERDES::MCA_SYND_KPX_SERDES. If MCA::KPX::SERDES::MCA_SYND_KPX_SERDES[ErrorPriority] is the same as the priority of the error in MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES, then the information in MCA::KPX::SERDES::MCA_SYND_KPX_SERDES is associated with the error in MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
46	<b>CECC.</b> Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor.

	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC</b> . Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred</b> . Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison</b> . Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub</b> . Reset: Cold,0. 1=The error was the result of a scrub operation.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId</b> . Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb</b> . Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[ErrorAddr]. A value of 0 indicates that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[55:0] contains a valid byte address. A value of 6 indicates that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[55:6] contains a valid cache line address and that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[55:12] contain a valid 4KB memory page and that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[11:0] should be ignored by error handling software.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt</b> . Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::KPX::SERDES::MCA_CTL_KPX_SERDES enables error reporting for the logged error.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode</b> . Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 74: MCA\_STATUS\_KPX\_SERDES

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
RAMECC	0x0	0/1	0/1	0/1	0	0	0
ARCIns	0x1	1	1	1	0	0	0
ARCData	0x2	1	1	1	0	0	0
APB	0x3	1	1	1	0	0	0

**MSR0000\_047E...MSRC000\_21F2 [KPX\_SERDES Machine Check Address]  
(MCA::KPX::SERDES::MCA\_ADDR\_KPX\_SERDES)**

Read-only. Reset: Cold,0000\_0000\_0000\_0000h.

MCA::KPX::SERDES::MCA\_ADDR\_KPX\_SERDES stores an address and other information associated with the error in MCA::KPX::SERDES::MCA\_STATUS\_KPX\_SERDES. The register is only meaningful if

MCA::KPX::SERDES::MCA\_STATUS\_KPX\_SERDES[Val]=1 and

MCA::KPX::SERDES::MCA\_STATUS\_KPX\_SERDES[AddrV]=1.

\_instKPXSERDES\_n[1:0]\_aliasMSRLEGACY; MSR0000\_047E

\_instKPXSERDES\_n[1:0]\_aliasMSR; MSRC000\_21F2

Bits	Description
63:0	<b>ErrorAddr.</b> Read-only. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES.

Table 75: MCA\_ADDR\_KPX\_SERDES

Error Type	Bits	Description
RAMECC	[63:0]	Reserved
ARCIns	[63:0]	Reserved
ARCData	[63:0]	Reserved
APB	[63:0]	Reserved

**MSR0000\_047F...MSRC000\_21F3 [KPX\_SERDES Machine Check Miscellaneous 0]  
(MCA::KPX::SERDES::MCA\_MISC0\_KPX\_SERDES)**

Log miscellaneous information associated with errors.

\_instKPXSERDES\_n[1:0]\_aliasMSRLEGACY; MSR0000\_047F

\_instKPXSERDES\_n[1:0]\_aliasMSR; MSRC000\_21F3

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   ! MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   ! MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   ! MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msrr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   ! MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   ! MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   ! MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_21F4 [KPX\_SERDES Machine Check Configuration]  
(MCA::KPX::SERDES::MCA\_CONFIG\_KPX\_SERDES)**

Reset: 0000\_0002\_0000\_0125h.

Controls configuration of the associated machine check bank.

\_instKPXSERDES\_n[1:0]\_aliasMSR; MSRC000\_21F4

Bits	Description
63:39	Reserved.
38:37	<b>DeferredIntType</b> . Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.
36:35	Reserved.
34	<b>LogDeferredInMcaStat</b> . Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES and MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES in addition to MCA::KPX::SERDES::MCA_DESTAT_KPX_SERDES and MCA::KPX::SERDES::MCA_DEADDR_KPX_SERDES. 0=Only log deferred errors in MCA::KPX::SERDES::MCA_DESTAT_KPX_SERDES and MCA::KPX::SERDES::MCA_DEADDR_KPX_SERDES. This bit does not affect logging of deferred errors in MCA::KPX::SERDES::MCA_SYND_KPX_SERDES, MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES.
33	Reserved.
32	<b>McaXEnable</b> . Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.
31:9	Reserved.
8	<b>McaLsbInStatusSupported</b> . Read-only. Reset: 1. 1=MCA::KPX::SERDES::MCA_CONFIG_KPX_SERDES[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.
5	<b>DeferredIntTypeSupported</b> . Read-only. Reset: 1. 1=MCA::KPX::SERDES::MCA_CONFIG_KPX_SERDES[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::KPX::SERDES::MCA_CONFIG_KPX_SERDES[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	<b>DeferredErrorLoggingSupported</b> . Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::KPX::SERDES::MCA_CONFIG_KPX_SERDES[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::KPX::SERDES::MCA_DESTAT_KPX_SERDES and MCA::KPX::SERDES::MCA_DEADDR_KPX_SERDES are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	<b>McaX</b> . Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::KPX::SERDES::MCA_MISC0_KPX_SERDES[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES[TCC] is present.

**MSRC000\_21F5 [KPX\_SERDES IP Identification] (MCA::KPX::SERDES::MCA\_IPID\_KPX\_SERDES)**

Reset: 0000\_0259\_0000\_0000h.

The MCA::KPX::SERDES::MCA\_IPID\_KPX\_SERDES register is used by software to determine what IP type and revision is associated with the MCA bank.

\_instKPIXSERDES\_n[1:0]\_aliasMSR; MSRC000\_21F5

Bits	Description
63:48	<b>McaType</b> . Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi</b> . Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID</b> . Read-only. Reset: 259h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId</b> . Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

**MSRC000\_21F6 [KPX\_SERDES Machine Check Syndrome] (MCA::KPX::SERDES::MCA\_SYND\_KPX\_SERDES)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::KPX::SERDES::MCA\_STATUS\_KPX\_SERDES Thread 0

\_instKPIXSERDES\_n[1:0]\_aliasMSR; MSRC000\_21F6

Bits	Description
63:33	Reserved.
32	<b>Syndrome</b> . Read-write, Volatile. Reset: Cold, 0. Contains the syndrome, if any, associated with the error logged in MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::KPX::SERDES::MCA_SYND_KPX_SERDES[Length]. The Syndrome field is only valid when MCA::KPX::SERDES::MCA_SYND_KPX_SERDES[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority</b> . Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::KPX::SERDES::MCA_SYND_KPX_SERDES. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length</b> . Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::KPX::SERDES::MCA_SYND_KPX_SERDES[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::KPX::SERDES::MCA_SYND_KPX_SERDES. For example, a syndrome length of 9 means that MCA::KPX::SERDES::MCA_SYND_KPX_SERDES[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation</b> . Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 76 [MCA_SYND_KPX_SERDES].

Table 76: MCA\_SYND\_KPX\_SERDES

Error Type	Bits	Description
RAMECC	[17:0]	Reserved
ARCIns	[17:0]	Reserved
ARCData	[17:0]	Reserved
APB	[17:0]	Reserved



**MSRC000\_21F8 [KPX\_SERDES Machine Check Deferred Error Status]  
(MCA::KPX::SERDES::MCA\_DESTAT\_KPX\_SERDES)**

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_instKPXSERDES\_n[1:0]\_aliasMSR; MSRC000\_21F8

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::KPX::SERDES::MCA_DEADDR_KPX_SERDES contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::KPX::SERDES::MCA_SYND_KPX_SERDES. If MCA::KPX::SERDES::MCA_SYND_KPX_SERDES[ErrorPriority] is the same as the priority of the error in MCA::KPX::SERDES::MCA_STATUS_KPX_SERDES, then the information in MCA::KPX::SERDES::MCA_SYND_KPX_SERDES is associated with the error in MCA::KPX::SERDES::MCA_DESTAT_KPX_SERDES.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[ErrorAddr]. A value of 0 indicates that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[55:0] contains a valid byte address. A value of 6 indicates that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[55:6] contains a valid cache line address and that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[55:12] contain a valid 4KB memory page and that MCA::KPX::SERDES::MCA_ADDR_KPX_SERDES[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC001\_041F [KPX\_SERDES Machine Check Control Mask]  
(MCA::KPX::SERDES::MCA\_CTL\_MASK\_KPX\_SERDES)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_instKPXSERDES\_n[1:0]\_aliasMSR; MSRC001\_041F

Bits	Description
63:4	Reserved.
3	<b>APB.</b> Read-write. Reset: 0. PHY APB error
2	<b>ARCData.</b> Read-write. Reset: 0. ARC data buffer parity error
1	<b>ARCIns.</b> Read-write. Reset: 0. ARC instruction buffer parity error
0	<b>RAMECC.</b> Read-write. Reset: 0. RAM ECC Error.

**3.2.6.14 KPX GMI****MSR0000\_0448...MSRC000\_2130 [KPX\_GMI Machine Check Control]  
(MCA::KPX::GMI::MCA\_CTL\_KPX\_GMI)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::KPX::GMI::MCA\_CTL\_KPX\_GMI register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSRLEGACY; MSR0000\_0448

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSRLEGACY; MSR0000\_044C

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSR; MSRC000\_2120

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_2130

Bits	Description
63:4	Reserved.
3	<b>APB.</b> Read-write. Reset: 0. PHY APB error
2	<b>ARCData.</b> Read-write. Reset: 0. ARC data buffer parity error
1	<b>ARCIns.</b> Read-write. Reset: 0. ARC instruction buffer parity error
0	<b>RAMECC.</b> Read-write. Reset: 0. RAM ECC Error.



**MSR0000\_0449...MSRC000\_2131 [KPX\_GMI Machine Check Status]  
(MCA::KPX::GMI::MCA\_STATUS\_KPX\_GMI)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSRLEGACY; MSR0000\_0449

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSRLEGACY; MSR0000\_044D

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSR; MSRC000\_2121

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_2131

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::KPX::GMI::MCA_CTL_KPX_GMI. This bit is a copy of bit in MCA::KPX::GMI::MCA_CTL_KPX_GMI for this error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::KPX::GMI::MCA_MISC0_KPX_GMI. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::KPX::GMI::MCA_ADDR_KPX_GMI contains address information associated with the error. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::KPX::GMI::MCA_STATUS_KPX_GMI[PCC]=0. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::KPX::GMI::MCA_SYND_KPX_GMI. If MCA::KPX::GMI::MCA_SYND_KPX_GMI[ErrorPriority] is the same as the priority of the error in MCA::KPX::GMI::MCA_STATUS_KPX_GMI, then the information in MCA::KPX::GMI::MCA_SYND_KPX_GMI is associated with the error in MCA::KPX::GMI::MCA_STATUS_KPX_GMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

46	<b>CECC</b> . Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC</b> . Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred</b> . Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison</b> . Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub</b> . Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId</b> . Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb</b> . Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::KPX::GMI::MCA_ADDR_KPX_GMI[ErrorAddr]. A value of 0 indicates that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[55:0] contains a valid byte address. A value of 6 indicates that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[55:6] contains a valid cache line address and that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[55:12] contain a valid 4KB memory page and that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[11:0] should be ignored by error handling software. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt</b> . Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::KPX::GMI::MCA_CTL_KPX_GMI enables error reporting for the logged error. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode</b> . Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 77: MCA\_STATUS\_KPX\_GMI

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
RAMECC	0x0	0/1	0/1	0/1	0	0	0
ARCIns	0x1	1	1	1	0	0	0
ARCData	0x2	1	1	1	0	0	0
APB	0x3	1	1	1	0	0	0

**MSR0000\_044A...MSRC000\_2132 [KPX\_GMI Machine Check Address]  
(MCA::KPX::GMI::MCA\_ADDR\_KPX\_GMI)**

Read-only. Reset: Cold,0000\_0000\_0000\_0000h.

MCA::KPX::GMI::MCA\_ADDR\_KPX\_GMI stores an address and other information associated with the error in MCA::KPX::GMI::MCA\_STATUS\_KPX\_GMI. The register is only meaningful if MCA::KPX::GMI::MCA\_STATUS\_KPX\_GMI[Val]=1 and MCA::KPX::GMI::MCA\_STATUS\_KPX\_GMI[AddrV]=1.

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSRLEGACY; MSR0000\_044A

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSRLEGACY; MSR0000\_044E

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSR; MSRC000\_2122

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_2132

Bits	Description
63:0	<b>ErrorAddr.</b> Read-only. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::KPX::GMI::MCA_STATUS_KPX_GMI.

Table 78: MCA\_ADDR\_KPX\_GMI

Error Type	Bits	Description
RAMECC	[63:0]	Reserved
ARCIns	[63:0]	Reserved
ARCDData	[63:0]	Reserved
APB	[63:0]	Reserved

**MSR0000\_044B...MSRC000\_2133 [KPX\_GMI Machine Check Miscellaneous 0]  
(MCA::KPX::GMI::MCA\_MISC0\_KPX\_GMI)**

Log miscellaneous information associated with errors.

\_inst[GMI\_CONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSRLEGACY; MSR0000\_044B

\_ccd[1:0]\_instGMI\_CONTAINER12KPXGMI\_n[3,1]\_aliasMSRLEGACY; MSR0000\_044F

\_inst[GMI\_CONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSR; MSRC000\_2123

\_ccd[1:0]\_instGMI\_CONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_2133

Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   ! MCA::KPX::GMI::MCA_MISC0_KPX_GMI[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   ! MCA::KPX::GMI::MCA_MISC0_KPX_GMI[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   ! MCA::KPX::GMI::MCA_MISC0_KPX_GMI[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   ! MCA::KPX::GMI::MCA_MISC0_KPX_GMI[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   ! MCA::KPX::GMI::MCA_MISC0_KPX_GMI[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn]   ! MCA::KPX::GMI::MCA_MISC0_KPX_GMI[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

**MSRC000\_21[2...3]4 [KPX\_GMI Machine Check Configuration]  
(MCA::KPX::GMI::MCA\_CONFIG\_KPX\_GMI)**

Controls configuration of the associated machine check bank.

\_inst[GMI\_CONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSR; MSRC000\_2124

\_ccd[1:0]\_instGMI\_CONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_2134

Bits	Description
63:41	Reserved.
40	<p><b>IntEn.</b> 1=When set, this bank will generate corrected error interrupts.</p> <p>AccessType:     _ccd0_instGMI_CONTAINER12KPXGMI_n1_aliasMSR                   _ccd1_instGMI_CONTAINER12KPXGMI_n3_aliasMSR                   Read-write</p> <p>AccessType:     _instGMI_CONTAINER0KPXGMI_n0_aliasMSR                   _instGMI_CONTAINER1KPXGMI_n2_aliasMSR                   Reserved</p> <p>Reset:     _ccd0_instGMI_CONTAINER12KPXGMI_n1_aliasMSR              _ccd1_instGMI_CONTAINER12KPXGMI_n3_aliasMSR              0</p> <p>Init:     _ccd0_instGMI_CONTAINER12KPXGMI_n1_aliasMSR             _ccd1_instGMI_CONTAINER12KPXGMI_n3_aliasMSR             BIOS,0</p>
39	Reserved.
38:37	<p><b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.</p>
36:35	Reserved.
34	<p><b>LogDeferredInMcaStat.</b> Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::KPX::GMI::MCA_STATUS_KPX_GMI and MCA::KPX::GMI::MCA_ADDR_KPX_GMI in addition to MCA::KPX::GMI::MCA_DESTAT_KPX_GMI and MCA::KPX::GMI::MCA_DEADDR_KPX_GMI. 0=Only log deferred errors in MCA::KPX::GMI::MCA_DESTAT_KPX_GMI and MCA::KPX::GMI::MCA_DEADDR_KPX_GMI. This bit does not affect logging of deferred errors in MCA::KPX::GMI::MCA_SYND_KPX_GMI, MCA::KPX::GMI::MCA_MISC0_KPX_GMI.</p>
33	Reserved.
32	<p><b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.</p>
31:11	Reserved.
10	<p><b>IntPresent.</b> 1=This bank can be configured to trigger a corrected error interrupt using MCA::KPX::GMI::MCA_CONFIG_KPX_GMI[IntEn].</p> <p>AccessType:     _ccd0_instGMI_CONTAINER12KPXGMI_n1_aliasMSR                   _ccd1_instGMI_CONTAINER12KPXGMI_n3_aliasMSR                   Read-only, Volatile</p> <p>AccessType:     _instGMI_CONTAINER0KPXGMI_n0_aliasMSR                   _instGMI_CONTAINER1KPXGMI_n2_aliasMSR                   Reserved</p> <p>Reset:     _ccd0_instGMI_CONTAINER12KPXGMI_n1_aliasMSR              _ccd1_instGMI_CONTAINER12KPXGMI_n3_aliasMSR              0</p>
9	<p><b>McaFruTextInMca.</b> 1=FruText is reported McaSynd1/McaSynd2 registers</p> <p>AccessType:     _ccd0_instGMI_CONTAINER12KPXGMI_n1_aliasMSR                   _ccd1_instGMI_CONTAINER12KPXGMI_n3_aliasMSR                   Read-write</p> <p>AccessType:     _instGMI_CONTAINER0KPXGMI_n0_aliasMSR                   _instGMI_CONTAINER1KPXGMI_n2_aliasMSR                   Reserved</p> <p>Reset:     _ccd0_instGMI_CONTAINER12KPXGMI_n1_aliasMSR              _ccd1_instGMI_CONTAINER12KPXGMI_n3_aliasMSR</p>

	0	
	Init:	_ccd0_instGMICONTAINER12KPXGMI_n1_aliasMSR
		_ccd1_instGMICONTAINER12KPXGMI_n3_aliasMSR
		BIOS,0
8		<b>McaLsbInStatusSupported.</b> Read-only. Reset: 1. 1=MCA::KPX::GMI::MCA_CONFIG_KPX_GMI[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6		Reserved.
5		<b>DeferredIntTypeSupported.</b> Read-only. Reset: 1. 1=MCA::KPX::GMI::MCA_CONFIG_KPX_GMI[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::KPX::GMI::MCA_CONFIG_KPX_GMI[DeferredErrorLoggingSupported]=1.
4:3		Reserved.
2		<b>DeferredErrorLoggingSupported.</b> Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::KPX::GMI::MCA_CONFIG_KPX_GMI[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::KPX::GMI::MCA_DESTAT_KPX_GMI and MCA::KPX::GMI::MCA_DEADDR_KPX_GMI are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1		Reserved.
0		<b>McaX.</b> Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::KPX::GMI::MCA_MISC0_KPX_GMI[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::KPX::GMI::MCA_STATUS_KPX_GMI[TCC] is present.

#### MSRC000\_21[2...3]5 [KPX\_GMI IP Identification] (MCA::KPX::GMI::MCA\_IPID\_KPX\_GMI)

Reset: 0000\_0269\_0000\_0000h.

The MCA::KPX::GMI::MCA\_IPID\_KPX\_GMI register is used by software to determine what IP type and revision is associated with the MCA bank.

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSR; MSRC000\_2125

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_2135

Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 269h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.



<b>MSRC000_21[2...3]6 [KPX_GMI Machine Check Syndrome] (MCA::KPX::GMI::MCA_SYND_KPX_GMI)</b>	
Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h.	
Logs physical location information associated with error in MCA::KPX::GMI::MCA_STATUS_KPX_GMI Thread 0	
_inst[GMICONTAINER[1:0]KPXGMI]_n[2,0]_aliasMSR; MSRC000_2126	
_ccd[1:0]_instGMICONTAINER12KPXGMI_n[3,1]_aliasMSR; MSRC000_2136	
<b>Bits</b>	<b>Description</b>
63:33	Reserved.
32	<b>Syndrone.</b> Read-write, Volatile. Reset: Cold, 0. Contains the syndrome, if any, associated with the error logged in MCA::KPX::GMI::MCA_STATUS_KPX_GMI. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::KPX::GMI::MCA_SYND_KPX_GMI[Length]. The Syndrome field is only valid when MCA::KPX::GMI::MCA_SYND_KPX_GMI[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority.</b> Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::KPX::GMI::MCA_SYND_KPX_GMI. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length.</b> Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::KPX::GMI::MCA_SYND_KPX_GMI[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::KPX::GMI::MCA_SYND_KPX_GMI. For example, a syndrome length of 9 means that MCA::KPX::GMI::MCA_SYND_KPX_GMI[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation.</b> Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 79 [MCA_SYND_KPX_GMI].

Table 79: MCA\_SYND\_KPX\_GMI

<b>Error Type</b>	<b>Bits</b>	<b>Description</b>
RAMECC	[17:0]	Reserved
ARCIns	[17:0]	Reserved
ARCData	[17:0]	Reserved
APB	[17:0]	Reserved

**MSRC000\_21[2...3]8 [KPX\_GMI Machine Check Deferred Error Status]  
(MCA::KPX::GMI::MCA\_DESTAT\_KPX\_GMI)**

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSR; MSRC000\_2128

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_2138

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::KPX::GMI::MCA_DEADDR_KPX_GMI contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::KPX::GMI::MCA_SYND_KPX_GMI. If MCA::KPX::GMI::MCA_SYND_KPX_GMI[ErrorPriority] is the same as the priority of the error in MCA::KPX::GMI::MCA_STATUS_KPX_GMI, then the information in MCA::KPX::GMI::MCA_SYND_KPX_GMI is associated with the error in MCA::KPX::GMI::MCA_DESTAT_KPX_GMI.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::KPX::GMI::MCA_ADDR_KPX_GMI[ErrorAddr]. A value of 0 indicates that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[55:0] contains a valid byte address. A value of 6 indicates that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[55:6] contains a valid cache line address and that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[55:12] contain a valid 4KB memory page and that MCA::KPX::GMI::MCA_ADDR_KPX_GMI[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC000\_2139 [KPX\_GMI Deferred Error Address] (MCA::KPX::GMI::MCA\_DEADDR\_KPX\_GMI)**

Read-only. Reset: Cold,0000\_0000\_0000\_0000h.

The MCA::KPX::GMI::MCA\_DEADDR\_KPX\_GMI register stores the address associated with the error in MCA::KPX::GMI::MCA\_DESTAT\_KPX\_GMI. The register is only meaningful if MCA::KPX::GMI::MCA\_DESTAT\_KPX\_GMI[Val]=1 and MCA::KPX::GMI::MCA\_DESTAT\_KPX\_GMI[AddrV]=1. The lowest valid bit of the address is defined by MCA::KPX::GMI::MCA\_DESTAT\_KPX\_GMI[AddrLsb].

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_2139

Bits	Description
63:0	<b>ErrorAddr.</b> Read-only. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::KPX::GMI::MCA_DESTAT_KPX_GMI.



**MSRC001\_041[2...3] [KPX\_GMI Machine Check Control Mask]  
(MCA::KPX::GMI::MCA\_CTL\_MASK\_KPX\_GMI)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_inst[GMICONTAINER[1:0]KPXGMI]\_n[2,0]\_aliasMSR; MSRC001\_0412\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC001\_0413

Bits	Description
63:4	Reserved.
3	<b>APB.</b> Read-write. Reset: 0. PHY APB error
2	<b>ARCData.</b> Read-write. Reset: 0. ARC data buffer parity error
1	<b>ARCIns.</b> Read-write. Reset: 0. ARC instruction buffer parity error
0	<b>RAMECC.</b> Read-write. Reset: 0. RAM ECC Error.

**MSRC000\_213E [KPX\_GMI Machine Check Syndrome Extended]  
(MCA::KPX::GMI::MCA\_SYND1\_KPX\_GMI)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::KPX::GMI::MCA\_STATUS\_KPX\_GMI Thread 0

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_213E

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::KPX::GMI::MCA_SYND1_KPX_GMI register stores information associated with the error in MCA::KPX::GMI::MCA_STATUS_KPX_GMI or MCA_DESTAT. The register is meaningful if MCA::KPX::GMI::MCA_STATUS_KPX_GMI[SyndV]=1. When MCA::KPX::GMI::MCA_CONFIG_KPX_GMI[McaFruTextInMca]=1, MCA::KPX::GMI::MCA_SYND1_KPX_GMI stores ASCII FruText associated with the error.

**MSRC000\_213F [KPX\_GMI Machine Check Syndrome Extended]  
(MCA::KPX::GMI::MCA\_SYND2\_KPX\_GMI)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::KPX::GMI::MCA\_STATUS\_KPX\_GMI Thread 0

\_ccd[1:0]\_instGMICONTAINER12KPXGMI\_n[3,1]\_aliasMSR; MSRC000\_213F

Bits	Description
63:0	<b>Syndrome.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::KPX::GMI::MCA_SYND2_KPX_GMI register stores information associated with the error in MCA::KPX::GMI::MCA_STATUS_KPX_GMI or MCA_DESTAT. The register is meaningful if MCA::KPX::GMI::MCA_STATUS_KPX_GMI[SyndV]=1. When MCA::KPX::GMI::MCA_CONFIG_KPX_GMI[McaFruTextInMca]=1, MCA::KPX::GMI::MCA_SYND2_KPX_GMI stores ASCII FruText associated with the error.

## 3.2.6.15 PCS GMI

**MSR0000\_0440...MSRC000\_2110 [PCS\_GMI Machine Check Control]  
(MCA::PCS::GMI::MCA\_CTL\_PCS\_GMI)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

0=Disables error reporting for the corresponding error. 1=Enables error reporting via machine check exception for the corresponding error. The MCA::PCS::GMI::MCA\_CTL\_PCS\_GMI register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG\_CTL. Does not affect error detection, correction, or logging.

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSRLEGACY; MSR0000\_0440

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSRLEGACY; MSR0000\_0444

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSR; MSRC000\_2100

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_2110

Bits	Description
63:32	Reserved.
31	<b>TwixDataLoss.</b> Read-write. Reset: 0. TwixDataLoss
30	<b>LFDSFcinitTimeoutErr.</b> Read-write. Reset: 0. LFDSFcinitTimeoutErr
29	<b>LFDSTrainingTimeoutErr.</b> Read-write. Reset: 0. LFDSTrainingTimeoutErr
28	<b>RxCMDPktErr.</b> Read-write. Reset: 0. RxCMDPktErr
27	<b>LinkSubRxTimeoutErr.</b> Read-write. Reset: 0. LinkSubRxTimeoutErr
26	<b>LinkSubTxTimeoutErr.</b> Read-write. Reset: 0. LinkSubTxTimeoutErr
25	<b>RxLfdsFifoUnderflowErr.</b> Read-write. Reset: 0. RxLfdsFifoUnderflowErr
24	<b>RxLfdsFifoOverflowErr.</b> Read-write. Reset: 0. RxLfdsFifoOverflowErr
23	<b>TwixRxBuff.</b> Read-write. Reset: 0. TwixRxBuff
22	<b>DeskewAbortErr.</b> Read-write. Reset: 0. DeskewAbortErr
21	<b>RecoveryRelockAttemptErr.</b> Read-write. Reset: 0. RecoveryRelockAttemptErr
20	<b>RecoveryAttemptErr.</b> Read-write. Reset: 0. RecoveryAttemptErr
19	<b>ReadySerialAttemptErr.</b> Read-write. Reset: 0. ReadySerialAttemptErr
18	<b>ReadySerialTimeoutErr.</b> Read-write. Reset: 0. ReadySerialTimeoutErr
17	<b>RecoveryTimeoutErr.</b> Read-write. Reset: 0. RecoveryTimeoutErr
16	<b>FCInitTimeoutErr.</b> Read-write. Reset: 0. FCInitTimeoutErr
15	<b>DataStartupLimitErr.</b> Read-write. Reset: 0. DataStartupLimitErr
14	<b>TwixOffline.</b> Read-write. Reset: 0. TwixOffline
13	<b>DeskewErr.</b> Read-write. Reset: 0. DeskewErr
12	<b>ElasticFifoOverflowErr.</b> Read-write. Reset: 0. ElasticFifoOverflowErr
11	<b>ReplayFifoUnderflowErr.</b> Read-write. Reset: 0. ReplayFifoUnderflowErr
10	<b>ReplayFifoOverflowErr.</b> Read-write. Reset: 0. ReplayFifoOverflowErr
9	<b>TxTwixOverflow.</b> Read-write. Reset: 0. TxTwixOverflow
8	<b>ReplayBufParityErr.</b> Read-write. Reset: 0. ReplayBufParityErr
7	<b>TxTwixFifoUnderflow.</b> Read-write. Reset: 0. TxTwixFifoUnderflow
6	<b>BERExceededErr.</b> Read-write. Reset: 0. BERExceededErr
5	<b>CRCErr.</b> Read-write. Reset: 0. CRCErr
4	<b>RxFifoOverflowErr.</b> Read-write. Reset: 0. RxFifoOverflowErr
3	<b>RxFifoUnderflowErr.</b> Read-write. Reset: 0. RxFifoUnderflowErr
2	<b>ReplayParityTwix.</b> Read-write. Reset: 0. ReplayParityTwix
1	<b>TrainingErr.</b> Read-write. Reset: 0. TrainingErr.
0	<b>DataLossErr.</b> Read-write. Reset: 0. DataLossErr

**MSR0000\_0441...MSRC000\_2111 [PCS\_GMI Machine Check Status]  
(MCA::PCS::GMI::MCA\_STATUS\_PCS\_GMI)**

Reset: Cold,0000\_0000\_0000\_0000h.

Logs information associated with errors.

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSRLEGACY; MSR0000\_0441

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSRLEGACY; MSR0000\_0445

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSR; MSRC000\_2101

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_2111

Bits	Description
63	<b>Val.</b> Reset: Cold,0. 1=A valid error has been detected. This bit should be cleared by software after the register has been read. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
62	<b>Overflow.</b> Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. See 3.1.3 [Machine Check Errors]. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
61	<b>UC.</b> Reset: Cold,0. 1=The error was not corrected by hardware. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
60	<b>En.</b> Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::PCS::GMI::MCA_CTL_PCS_GMI. This bit is a copy of bit in MCA::PCS::GMI::MCA_CTL_PCS_GMI for this error. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	<b>MiscV.</b> Reset: Cold,0. 1=Valid thresholding in MCA::PCS::GMI::MCA_MISC0_PCS_GMI. In certain modes, MISC registers are owned by platform firmware and will RAZ when read by non-SMM code. Therefore, it is possible for MiscV=1 and the MISC register to read as all zeros. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	<b>AddrV.</b> Reset: Cold,0. 1=MCA::PCS::GMI::MCA_ADDR_PCS_GMI contains address information associated with the error. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	<b>PCC.</b> Reset: Cold,0. 1=Hardware context held by the processor may have been corrupted. Continued operation of the system may have unpredictable results. The error is not recoverable or survivable, and the system should be reinitialized. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
56	<b>ErrCoreIdVal.</b> Reset: Cold,0. 1=The ErrCoreId field is valid. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
55	<b>TCC.</b> Reset: Cold,0. 1=Hardware context of the process thread to which the error was reported may have been corrupted. Continued operation of the thread may have unpredictable results. The thread must be terminated. Only meaningful when MCA::PCS::GMI::MCA_STATUS_PCS_GMI[PCC]=0. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	<b>RESERV54.</b> Reset: Cold,0. MCA_STATUS Register Reserved bit. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
53	<b>SyndV.</b> Reset: Cold,0. 1=This error logged information in MCA::PCS::GMI::MCA_SYND_PCS_GMI. If MCA::PCS::GMI::MCA_SYND_PCS_GMI[ErrorPriority] is the same as the priority of the error in MCA::PCS::GMI::MCA_STATUS_PCS_GMI, then the information in MCA::PCS::GMI::MCA_SYND_PCS_GMI is associated with the error in MCA::PCS::GMI::MCA_STATUS_PCS_GMI. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52	Reserved.
51:47	<b>RESERV47.</b> Reset: Cold,00h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

46	<b>CECC</b> . Reset: Cold,0. 1=The error was a correctable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
45	<b>UECC</b> . Reset: Cold,0. 1=The error was an uncorrectable ECC error according to the restrictions of the ECC algorithm. UC indicates whether the error was actually corrected by the processor. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
44	<b>Deferred</b> . Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; an exception is deferred until the erroneous data is consumed. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
43	<b>Poison</b> . Reset: Cold,0. 1=The error was the result of attempting to consume poisoned data. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
42:41	<b>RESERV41</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
40	<b>Scrub</b> . Reset: Cold,0. 1=The error was the result of a scrub operation. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
39:38	<b>RESERV38</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
37:32	<b>ErrCoreId</b> . Reset: Cold,00h. When ErrCoreIdVal=1 this field indicates which core within the processor is associated with the error; Otherwise this field is reserved. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
31:30	<b>RESERV30</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
29:24	<b>AddrLsb</b> . Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::PCS::GMI::MCA_ADDR_PCS_GMI[ErrorAddr]. A value of 0 indicates that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[55:0] contains a valid byte address. A value of 6 indicates that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[55:6] contains a valid cache line address and that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[55:12] contain a valid 4KB memory page and that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[11:0] should be ignored by error handling software. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
23:22	<b>RESERV22</b> . Reset: Cold,0h. MCA_STATUS Register Reserved bits. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
21:16	<b>ErrorCodeExt</b> . Reset: Cold,00h. Extended Error Code. This field is used to identify the error type for root cause analysis. This field indicates which bit position in MCA::PCS::GMI::MCA_CTL_PCS_GMI enables error reporting for the logged error. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	<b>ErrorCode</b> . Reset: Cold,0000h. Error code for this error. See 3.1.3.3 [Error Codes] for details on decoding this field. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

Table 80: MCA\_STATUS\_PCS\_GMI

Error Type	ErrorCode Ext	UC	PCC	TCC	Deferred	Poison	AddrV
DataLossErr	0x0	0	0	0	0	0	0
TrainingErr	0x1	0	0	0	0	0	0
ReplayParity Twix	0x2	0	0	0	0	0	0

RxFifoUnderflowErr	0x3	0	0	0	0	0	0
RxFifoOverflowErr	0x4	0	0	0	0	0	0
CRCErr	0x5	0	0	0	0	0	0
BERExceededErr	0x6	0	0	0	0	0	0
TxTwixFifoUnderflow	0x7	0	0	0	0	0	0
ReplayBufParityErr	0x8	0	0	0	0	0	0
TxTwixOverflow	0x9	0	0	0	0	0	0
ReplayFifoOverflowErr	0xa	0	0	0	0	0	0
ReplayFifoUnderflowErr	0xb	0	0	0	0	0	0
ElasticFifoOverflowErr	0xc	0	0	0	0	0	0
DeskewErr	0xd	0	0	0	0	0	0
TwixOffline	0xe	0	0	0	0	0	0
DataStartupLimitErr	0xf	0	0	0	0	0	0
FCInitTimeoutErr	0x10	0	0	0	0	0	0
RecoveryTimeoutErr	0x11	0	0	0	0	0	0
ReadySerialTimeoutErr	0x12	0	0	0	0	0	0
ReadySerialAttemptErr	0x13	0	0	0	0	0	0
RecoveryAttemptErr	0x14	0	0	0	0	0	0
RecoveryRelockAttemptErr	0x15	0	0	0	0	0	0
DeskewAbortErr	0x16	0	0	0	0	0	0
TwixRxBuff	0x17	0	0	0	0	0	0
RxLfdsFifoOverflowErr	0x18	0	0	0	0	0	0
RxLfdsFifoUnderflowErr	0x19	0	0	0	0	0	0
LinkSubTxTimeoutErr	0x1a	0	0	0	0	0	0
LinkSubRxTimeoutErr	0x1b	0	0	0	0	0	0
RxCMDPktErr	0x1c	0	0	0	0	0	0

LFDSTrainin gTimeoutErr	0x1d	0	0	0	0	0	0
LFDSFcinitT imeoutErr	0x1e	0	0	0	0	0	0
TwixDataLo ss	0x1f	0	0	0	0	0	0

#### MSR0000\_0442...MSRC000\_2112 [PCS\_GMI Machine Check Address] (MCA::PCS::GMI::MCA\_ADDR\_PCS\_GMI)

Read-only. Reset: Cold,0000\_0000\_0000\_0000h.

MCA::PCS::GMI::MCA\_ADDR\_PCS\_GMI stores an address and other information associated with the error in MCA::PCS::GMI::MCA\_STATUS\_PCS\_GMI. The register is only meaningful if MCA::PCS::GMI::MCA\_STATUS\_PCS\_GMI[Val]=1 and MCA::PCS::GMI::MCA\_STATUS\_PCS\_GMI[AddrV]=1.

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSRLEGACY; MSR0000\_0442

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSRLEGACY; MSR0000\_0446

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSR; MSRC000\_2102

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_2112

Bits	Description
63:0	<b>ErrorAddr.</b> Read-only. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::PCS::GMI::MCA_STATUS_PCS_GMI.

Table 81: MCA\_ADDR\_PCS\_GMI

Error Type	Bits	Description
DataLossErr	[63:0]	Reserved
TrainingErr	[63:0]	Reserved
ReplayParityTwix	[63:0]	Reserved
RxFifoUnderflowErr	[63:0]	Reserved
RxFifoOverflowErr	[63:0]	Reserved
CRCErr	[63:0]	Reserved
BERExceededErr	[63:0]	Reserved
TxTwixFifoUnderflow	[63:0]	Reserved
ReplayBufParityErr	[63:0]	Reserved
TxTwixOverflow	[63:0]	Reserved
ReplayFifoOverflowErr	[63:0]	Reserved
ReplayFifoUnderflowErr	[63:0]	Reserved
ElasticFifoOverflowErr	[63:0]	Reserved
DeskewErr	[63:0]	Reserved
TwixOffline	[63:0]	Reserved
DataStartupLimitErr	[63:0]	Reserved
FCInitTimeoutErr	[63:0]	Reserved
RecoveryTimeoutErr	[63:0]	Reserved
ReadySerialTimeoutErr	[63:0]	Reserved
ReadySerialAttemptErr	[63:0]	Reserved
RecoveryAttemptErr	[63:0]	Reserved
RecoveryRelockAttemptErr	[63:0]	Reserved
DeskewAbortErr	[63:0]	Reserved
TwixRxBuff	[63:0]	Reserved
RxLfdsFifoOverflowErr	[63:0]	Reserved
RxLfdsFifoUnderflowErr	[63:0]	Reserved
LinkSubTxTimeoutErr	[63:0]	Reserved

LinkSubRxTimeoutErr	[63:0]	Reserved
RxCMDPktErr	[63:0]	Reserved
LFDSTrainingTimeoutErr	[63:0]	Reserved
LFDSFcinitTimeoutErr	[63:0]	Reserved
TwixDataLoss	[63:0]	Reserved



**MSR0000\_0443...MSRC000\_2113 [PCS\_GMI Machine Check Miscellaneous 0]  
(MCA::PCS::GMI::MCA\_MISC0\_PCS\_GMI)**

Log miscellaneous information associated with errors.	
_inst[GMICONTAINER[1:0]PCSGMI]_n[2,0]_aliasMSRLEGACY; MSR0000_0443	
_ccd[1:0]_instGMICONTAINER12PCSGMI_n[3,1]_aliasMSRLEGACY; MSR0000_0447	
_inst[GMICONTAINER[1:0]PCSGMI]_n[2,0]_aliasMSR; MSRC000_2103	
_ccd[1:0]_instGMICONTAINER12PCSGMI_n[3,1]_aliasMSR; MSRC000_2113	
Bits	Description
63	<b>Valid.</b> Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	<b>CntP.</b> Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	<b>Locked.</b> Reset: 0. 1=Writes to this register are ignored. This bit is set by BIOS to indicate that this register is not available for OS use. BIOS should set this bit if ThresholdIntType is set to SMI. AccessType: Core::X86::Msrr::HWCR[McStatusWrEn] ? Read-write : Read-only.
60	<b>IntP.</b> Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::PCS::GMI::MCA_MISC0_PCS_GMI[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	<b>LvtOffset.</b> Reset: 0h. One per die. For error thresholding interrupts, specifies the address of the LVT entry in the APIC registers as follows: LVT address = (LvtOffset shifted left 4 bits) + 500h (see Core::X86::Apic::ExtendedInterruptLvtEntries). AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::PCS::GMI::MCA_MISC0_PCS_GMI[Locked]) ? Read-write : Read-only.
51	<b>CntEn.</b> Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::PCS::GMI::MCA_MISC0_PCS_GMI[Locked]) ? Read-write : Read-only.
50:49	<b>ThresholdIntType.</b> Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set and IntP==1. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msrr::McaIntrCfg[ThresholdLvtOffset]). 10b = SMI trigger event. 11b = Reserved. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::PCS::GMI::MCA_MISC0_PCS_GMI[Locked]) ? Read-write : Read-only.
48	<b>Ovrflw.</b> Reset: Cold,0. Set by hardware when ErrCnt transitions from FFEh to FFFh. When this field is set, ErrCnt no longer increments. When this bit is set, the interrupt selected by the ThresholdIntType field is generated. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::PCS::GMI::MCA_MISC0_PCS_GMI[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	<b>ErrCnt.</b> Reset: Cold,000h. This is written by software to set the starting value of the error counter. This is incremented by hardware when errors are logged. When this counter overflows, it stays at FFFh (no rollover). The threshold value, written by software, is (FFFh - the desired error count (the number of errors necessary in order for an interrupt to be taken)); the desired error count of 0 (a write value of FFFh) is not supported. AccessType: (Core::X86::Msrr::HWCR[McStatusWrEn]   !MCA::PCS::GMI::MCA_MISC0_PCS_GMI[Locked]) ? Read-write : Read-only.
31:24	<b>BlkPtr.</b> Read-write. Reset: 00h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.



**MSRC000\_21[0...1]4 [PCS\_GMI Machine Check Configuration]  
(MCA::PCS::GMI::MCA\_CONFIG\_PCS\_GMI)**

Controls configuration of the associated machine check bank.

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSR; MSRC000\_2104

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_2114

Bits	Description
63:41	Reserved.
40	<p><b>IntEn.</b> 1=When set, this bank will generate corrected error interrupts.</p> <p>AccessType:     _ccd0_instGMICONTAINER12PCSGMI_n1_aliasMSR                   _ccd1_instGMICONTAINER12PCSGMI_n3_aliasMSR                   Read-write</p> <p>AccessType:     _instGMICONTAINER0PCSGMI_n0_aliasMSR                   _instGMICONTAINER1PCSGMI_n2_aliasMSR                   Reserved</p> <p>Reset:     _ccd0_instGMICONTAINER12PCSGMI_n1_aliasMSR             _ccd1_instGMICONTAINER12PCSGMI_n3_aliasMSR             0</p> <p>Init:     _ccd0_instGMICONTAINER12PCSGMI_n1_aliasMSR            _ccd1_instGMICONTAINER12PCSGMI_n3_aliasMSR            BIOS,0</p>
39	Reserved.
38:37	<p><b>DeferredIntType.</b> Read-write. Reset: 0h. Specifies the type of interrupt signaled when a deferred error is logged. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[DeferredLvtOffset]). 10b = SMI trigger event. 11b = Reserved.</p>
36:35	Reserved.
34	<p><b>LogDeferredInMcaStat.</b> Read-write. Reset: 0. Init: BIOS,1. 1=Log deferred errors in MCA::PCS::GMI::MCA_STATUS_PCS_GMI and MCA::PCS::GMI::MCA_ADDR_PCS_GMI in addition to MCA::PCS::GMI::MCA_DESTAT_PCS_GMI and MCA::PCS::GMI::MCA_DEADDR_PCS_GMI. 0=Only log deferred errors in MCA::PCS::GMI::MCA_DESTAT_PCS_GMI and MCA::PCS::GMI::MCA_DEADDR_PCS_GMI. This bit does not affect logging of deferred errors in MCA::PCS::GMI::MCA_SYND_PCS_GMI, MCA::PCS::GMI::MCA_MISC0_PCS_GMI.</p>
33	Reserved.
32	<p><b>McaXEnable.</b> Read-write. Reset: 0. Init: BIOS,1. Check: 1. 1=Software has acknowledged support for the MCAX feature set. 0=Software has not acknowledged support for the MCAX feature set. All uncorrected and fatal errors will cause an ErrorEvent packet to be generated. Deferred error interrupts are configured via Core::X86::Msr::McaIntrCfg.</p>
31:11	Reserved.
10	<p><b>IntPresent.</b> 1=This bank can be configured to trigger a corrected error interrupt using MCA::PCS::GMI::MCA_CONFIG_PCS_GMI[IntEn].</p> <p>AccessType:     _ccd0_instGMICONTAINER12PCSGMI_n1_aliasMSR                   _ccd1_instGMICONTAINER12PCSGMI_n3_aliasMSR                   Read-only, Volatile</p> <p>AccessType:     _instGMICONTAINER0PCSGMI_n0_aliasMSR                   _instGMICONTAINER1PCSGMI_n2_aliasMSR                   Reserved</p> <p>Reset:     _ccd0_instGMICONTAINER12PCSGMI_n1_aliasMSR             _ccd1_instGMICONTAINER12PCSGMI_n3_aliasMSR             0</p>
9	<p><b>McaFruTextInMca.</b> 1=FruText is reported McaSynd1/McaSynd2 registers</p> <p>AccessType:     _ccd0_instGMICONTAINER12PCSGMI_n1_aliasMSR                   _ccd1_instGMICONTAINER12PCSGMI_n3_aliasMSR                   Read-write</p> <p>AccessType:     _instGMICONTAINER0PCSGMI_n0_aliasMSR                   _instGMICONTAINER1PCSGMI_n2_aliasMSR                   Reserved</p> <p>Reset:     _ccd0_instGMICONTAINER12PCSGMI_n1_aliasMSR             _ccd1_instGMICONTAINER12PCSGMI_n3_aliasMSR</p>

	0	
	Init:	_ccd0_instGMICONTAINER12PCSGMI_n1_aliasMSR
		_ccd1_instGMICONTAINER12PCSGMI_n3_aliasMSR
		BIOS,0
8	<b>McaLsbInStatusSupported.</b>	Read-only. Reset: 1. 1=MCA::PCS::GMI::MCA_CONFIG_PCS_GMI[McaLsbInStatusSupported] indicates that AddrLsb is located in McaStatus registers.
7:6	Reserved.	
5	<b>DeferredIntTypeSupported.</b>	Read-only. Reset: 1. 1=MCA::PCS::GMI::MCA_CONFIG_PCS_GMI[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::PCS::GMI::MCA_CONFIG_PCS_GMI[DeferredErrorLoggingSupported]=1.
4:3	Reserved.	
2	<b>DeferredErrorLoggingSupported.</b>	Read-only. Reset: 1. 1=Deferred errors are supported in this MCA bank, and MCA::PCS::GMI::MCA_CONFIG_PCS_GMI[LogDeferredInMcaStat] controls the logging behavior of these errors. MCA::PCS::GMI::MCA_DESTAT_PCS_GMI and MCA::PCS::GMI::MCA_DEADDR_PCS_GMI are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.	
0	<b>McaX.</b>	Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::PCS::GMI::MCA_MISC0_PCS_GMI[BlkPtr] indicates the presence of the additional MISC registers, but is not used to determine their MSR numbers. Deferred error interrupt type is specifiable by MCA bank. MCA::PCS::GMI::MCA_STATUS_PCS_GMI[TCC] is present.

#### MSRC000\_21[0...1]5 [PCS\_GMI IP Identification] (MCA::PCS::GMI::MCA\_IPID\_PCS\_GMI)

Reset: 0000\_0241\_0000\_0000h.

The MCA::PCS::GMI::MCA\_IPID\_PCS\_GMI register is used by software to determine what IP type and revision is associated with the MCA bank.

\_inst[GMICONTAINER[1:0]PCSGMI\_n[2,0]\_aliasMSR; MSRC000\_2105

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_2115

Bits	Description
63:48	<b>McaType.</b> Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	<b>InstanceIdHi.</b> Read-write. Reset: 0h. The HI value instance ID of this IP. This is initialized to a unique ID per instance of this register.
43:32	<b>HardwareID.</b> Read-only. Reset: 241h. The Hardware ID of the IP associated with this MCA bank.
31:0	<b>InstanceId.</b> Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

<b>MSRC000_21[0...1]6 [PCS_GMI Machine Check Syndrome] (MCA::PCS::GMI::MCA_SYND_PCS_GMI)</b>	
Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h.	
Logs physical location information associated with error in MCA::PCS::GMI::MCA_STATUS_PCS_GMI Thread 0	
_inst[GMICONTAINER[1:0]PCSGMI]_n[2,0]_aliasMSR; MSRC000_2106	
_ccd[1:0]_instGMICONTAINER12PCSGMI_n[3,1]_aliasMSR; MSRC000_2116	
<b>Bits</b>	<b>Description</b>
63:32	<b>Syndrom</b> . Read-write, Volatile. Reset: Cold, 0000_0000h. Contains the syndrome, if any, associated with the error logged in MCA::PCS::GMI::MCA_STATUS_PCS_GMI. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::PCS::GMI::MCA_SYND_PCS_GMI[Length]. The Syndrome field is only valid when MCA::PCS::GMI::MCA_SYND_PCS_GMI[Length] is not 0.
31:27	Reserved.
26:24	<b>ErrorPriority</b> . Read-write, Volatile. Reset: Cold, 0h. Encodes the priority of the error logged in MCA::PCS::GMI::MCA_SYND_PCS_GMI. 3'b000 = No error; 3'b001 = Reserved; 3'b010 = Corrected Error; 3'b011 = Deferred Error; 3'b100 = Uncorrected Error; 3'b101 = Fatal Error; all others reserved.
23:18	<b>Length</b> . Read-write, Volatile. Reset: Cold, 00h. Specifies the length in bits of the syndrome contained in MCA::PCS::GMI::MCA_SYND_PCS_GMI[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::PCS::GMI::MCA_SYND_PCS_GMI. For example, a syndrome length of 9 means that MCA::PCS::GMI::MCA_SYND_PCS_GMI[Syndrome] bits [8:0] contains a valid syndrome.
17:0	<b>ErrorInformation</b> . Read-write, Volatile. Reset: Cold, 0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 82 [MCA_SYND_PCS_GMI].

Table 82: MCA\_SYND\_PCS\_GMI

<b>Error Type</b>	<b>Bits</b>	<b>Description</b>
DataLossErr	[17:0]	Reserved
TrainingErr	[17:0]	Reserved
ReplayParityTwix	[17:0]	Reserved
RxFifoUnderflowErr	[17:0]	Reserved
RxFifoOverflowErr	[17:0]	Reserved
CRCErr	[17:0]	Reserved
BERExceededErr	[17:0]	Reserved
TxTwixFifoUnderflow	[17:0]	Reserved
ReplayBufParityErr	[17:0]	Reserved
TxTwixOverflow	[17:0]	Reserved
ReplayFifoOverflowErr	[17:0]	Reserved
ReplayFifoUnderflowErr	[17:0]	Reserved
ElasticFifoOverflowErr	[17:0]	Reserved
DeskewErr	[17:0]	Reserved
TwixOffline	[17:0]	Reserved
DataStartupLimitErr	[17:0]	Reserved
FCInitTimeoutErr	[17:0]	Reserved
RecoveryTimeoutErr	[17:0]	Reserved
ReadySerialTimeoutErr	[17:0]	Reserved
ReadySerialAttemptErr	[17:0]	Reserved
RecoveryAttemptErr	[17:0]	Reserved
RecoveryRelockAttemptErr	[17:0]	Reserved
DeskewAbortErr	[17:0]	Reserved
TwixRxBuff	[17:0]	Reserved
RxLfdsFifoOverflowErr	[17:0]	Reserved
RxLfdsFifoUnderflowErr	[17:0]	Reserved

LinkSubTxTimeoutErr	[17:0]	Reserved
LinkSubRxTimeoutErr	[17:0]	Reserved
RxCMDPktErr	[17:0]	Reserved
LFDSTrainingTimeoutErr	[17:0]	Reserved
LFDSCfinitTimeoutErr	[17:0]	Reserved
TwixDataLoss	[17:0]	Reserved

#### MSRC000\_21[0...1]8 [PCS\_GMI Machine Check Deferred Error Status] (MCA::PCS::GMI::MCA\_DESTAT\_PCS\_GMI)

Reset: Cold,0000\_0000\_0000\_0000h.

Holds status information for the first deferred error seen in this bank.

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSR; MSRC000\_2108

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_2118

Bits	Description
63	<b>Val.</b> Read-write, Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	<b>Overflow.</b> Read-write, Volatile. Reset: Cold,0. 1=An error was detected while the valid bit (Val) was set; at least one error was not logged. Overflow is set independently of whether the existing error is overwritten. (See the section on overwrite priorities.)
61:59	<b>RESERV4.</b> Read-write. Reset: Cold,0h.
58	<b>AddrV.</b> Read-write, Volatile. Reset: Cold,0. 1=MCA::PCS::GMI::MCA_DEADDR_PCS_GMI contains address information associated with the error.
57:54	<b>RESERV3.</b> Read-write. Reset: Cold,0h.
53	<b>SyndV.</b> Read-write, Volatile. Reset: Cold,0. 1=This error logged information in MCA::PCS::GMI::MCA_SYND_PCS_GMI. If MCA::PCS::GMI::MCA_SYND_PCS_GMI[ErrorPriority] is the same as the priority of the error in MCA::PCS::GMI::MCA_STATUS_PCS_GMI, then the information in MCA::PCS::GMI::MCA_SYND_PCS_GMI is associated with the error in MCA::PCS::GMI::MCA_DESTAT_PCS_GMI.
52:45	<b>RESERV2.</b> Read-write. Reset: Cold,00h.
44	<b>Deferred.</b> Read-write, Volatile. Reset: Cold,0. 1=A deferred error was created. A deferred error is the result of an uncorrectable data error which did not immediately cause a processor exception; poison is created and an exception is deferred until the poison data is consumed.
43:30	<b>RESERV1.</b> Read-write. Reset: Cold,0000h.
29:24	<b>AddrLsb.</b> Read-write, Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::PCS::GMI::MCA_ADDR_PCS_GMI[ErrorAddr]. A value of 0 indicates that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[55:0] contains a valid byte address. A value of 6 indicates that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[55:6] contains a valid cache line address and that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[55:12] contain a valid 4KB memory page and that MCA::PCS::GMI::MCA_ADDR_PCS_GMI[11:0] should be ignored by error handling software.
23:22	<b>RESERV0.</b> Read-write. Reset: Cold,0h.
21:16	<b>ErrorCodeExt.</b> Read-write, Volatile. Reset: Cold,00h. Logs an extended error code when an error is detected. This model-specific field is used in conjunction with ErrorCode? to identify the error sub-type for root cause analysis.
15:0	<b>ErrorCode.</b> Read-write, Volatile. Reset: Cold,0000h. Error code for this error.

**MSRC000\_2119 [PCS\_GMI Deferred Error Address] (MCA::PCS::GMI::MCA\_DEADDR\_PCS\_GMI)**

Read-only. Reset: Cold,0000\_0000\_0000\_0000h.

The MCA::PCS::GMI::MCA\_DEADDR\_PCS\_GMI register stores the address associated with the error in MCA::PCS::GMI::MCA\_DESTAT\_PCS\_GMI. The register is only meaningful if MCA::PCS::GMI::MCA\_DESTAT\_PCS\_GMI[Val]=1 and MCA::PCS::GMI::MCA\_DESTAT\_PCS\_GMI[AddrV]=1. The lowest valid bit of the address is defined by MCA::PCS::GMI::MCA\_DESTAT\_PCS\_GMI[AddrLsb].

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_2119

Bits	Description
63:0	<b>ErrorAddr.</b> Read-only. Reset: Cold,0000_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::PCS::GMI::MCA_DESTAT_PCS_GMI.

**MSRC001\_041[0...1] [PCS\_GMI Machine Check Control Mask]  
(MCA::PCS::GMI::MCA\_CTL\_MASK\_PCS\_GMI)**

Read-write. Reset: 0000\_0000\_0000\_0000h.

Inhibit detection of an error source.

\_inst[GMICONTAINER[1:0]PCSGMI]\_n[2,0]\_aliasMSR; MSRC001\_0410

\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC001\_0411

Bits	Description
------	-------------

63:32	Reserved.
-------	-----------

31	<b>TwixDataLoss.</b> Read-write. Reset: 0. TwixDataLoss
----	---

30	<b>LFDSFcinitTimeoutErr.</b> Read-write. Reset: 0. LFDSFcinitTimeoutErr
----	---

29	<b>LFDSTrainingTimeoutErr.</b> Read-write. Reset: 0. LFDSTrainingTimeoutErr
----	---

28	<b>RxCMDPktErr.</b> Read-write. Reset: 0. RxCMDPktErr
----	---

27	<b>LinkSubRxTimeoutErr.</b> Read-write. Reset: 0. LinkSubRxTimeoutErr
----	---

26	<b>LinkSubTxTimeoutErr.</b> Read-write. Reset: 0. LinkSubTxTimeoutErr
----	---

25	<b>RxLfdsFifoUnderflowErr.</b> Read-write. Reset: 0. RxLfdsFifoUnderflowErr
----	---

24	<b>RxLfdsFifoOverflowErr.</b> Read-write. Reset: 0. RxLfdsFifoOverflowErr
----	---

23	<b>TwixRxBuff.</b> Read-write. Reset: 0. TwixRxBuff
----	---

22	<b>DeskewAbortErr.</b> Read-write. Reset: 0. DeskewAbortErr
----	---

21	<b>RecoveryRelockAttemptErr.</b> Read-write. Reset: 0. RecoveryRelockAttemptErr
----	---

20	<b>RecoveryAttemptErr.</b> Read-write. Reset: 0. RecoveryAttemptErr
----	---

19	<b>ReadySerialAttemptErr.</b> Read-write. Reset: 0. ReadySerialAttemptErr
----	---

18	<b>ReadySerialTimeoutErr.</b> Read-write. Reset: 0. ReadySerialTimeoutErr
----	---

17	<b>RecoveryTimeoutErr.</b> Read-write. Reset: 0. RecoveryTimeoutErr
----	---

16	<b>FCInitTimeoutErr.</b> Read-write. Reset: 0. FCInitTimeoutErr
----	---

15	<b>DataStartupLimitErr.</b> Read-write. Reset: 0. DataStartupLimitErr
----	---

14	<b>TwixOffline.</b> Read-write. Reset: 0. TwixOffline
----	---

13	<b>DeskewErr.</b> Read-write. Reset: 0. DeskewErr
----	---

12	<b>ElasticFifoOverflowErr.</b> Read-write. Reset: 0. ElasticFifoOverflowErr
----	---

11	<b>ReplayFifoUnderflowErr.</b> Read-write. Reset: 0. ReplayFifoUnderflowErr
----	---

10	<b>ReplayFifoOverflowErr.</b> Read-write. Reset: 0. ReplayFifoOverflowErr
----	---

9	<b>TxTwixOverflow.</b> Read-write. Reset: 0. TxTwixOverflow
---	---

8	<b>ReplayBufParityErr.</b> Read-write. Reset: 0. ReplayBufParityErr
---	---

7	<b>TxTwixFifoUnderflow.</b> Read-write. Reset: 0. TxTwixFifoUnderflow
---	---

6	<b>BERExceededErr.</b> Read-write. Reset: 0. BERExceededErr
---	---

5	<b>CRCErr.</b> Read-write. Reset: 0. CRCErr
---	---

4	<b>RxFifoOverflowErr.</b> Read-write. Reset: 0. RxFifoOverflowErr
---	---

3	<b>RxFifoUnderflowErr.</b> Read-write. Reset: 0. RxFifoUnderflowErr
---	---

2	<b>ReplayParityTwix.</b> Read-write. Reset: 0. ReplayParityTwix
---	---

1	<b>TrainingErr.</b> Read-write. Reset: 0. TrainingErr.
---	--

0	<b>DataLossErr.</b> Read-write. Reset: 0. DataLossErr
---	---

**MSRC000\_211E [PCS\_GMI Machine Check Syndrome Extended]  
(MCA::PCS::GMI::MCA\_SYND1\_PCS\_GMI)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::PCS::GMI::MCA\_STATUS\_PCS\_GMI Thread 0  
\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_211E

Bits	Description
63:0	<b>Syndrone.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::PCS::GMI::MCA_SYND1_PCS_GMI register stores information associated with the error in MCA::PCS::GMI::MCA_STATUS_PCS_GMI or MCA_DESTAT. The register is meaningful if MCA::PCS::GMI::MCA_STATUS_PCS_GMI[SyndV]=1. When MCA::PCS::GMI::MCA_CONFIG_PCS_GMI[McaFruTextInMca]=1, MCA::PCS::GMI::MCA_SYND1_PCS_GMI stores ASCII FruText associated with the error.

**MSRC000\_211F [PCS\_GMI Machine Check Syndrome Extended]  
(MCA::PCS::GMI::MCA\_SYND2\_PCS\_GMI)**

Read-write, Volatile. Reset: Cold, 0000\_0000\_0000\_0000h.

Logs physical location information associated with error in MCA::PCS::GMI::MCA\_STATUS\_PCS\_GMI Thread 0  
\_ccd[1:0]\_instGMICONTAINER12PCSGMI\_n[3,1]\_aliasMSR; MSRC000\_211F

Bits	Description
63:0	<b>Syndrone.</b> Read-write, Volatile. Reset: Cold, 0000_0000_0000_0000h. The MCA::PCS::GMI::MCA_SYND2_PCS_GMI register stores information associated with the error in MCA::PCS::GMI::MCA_STATUS_PCS_GMI or MCA_DESTAT. The register is meaningful if MCA::PCS::GMI::MCA_STATUS_PCS_GMI[SyndV]=1. When MCA::PCS::GMI::MCA_CONFIG_PCS_GMI[McaFruTextInMca]=1, MCA::PCS::GMI::MCA_SYND2_PCS_GMI stores ASCII FruText associated with the error.

## 4 Advanced Platform Management Link (APML)

### 4.1 Overview

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

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

#### 4.1.1 Definitions

Table 83: APML Definitions

Term	Description
<b>ARA</b>	Alert response address.
<b>ARP</b>	Address Resolution Protocol
<b>EC</b>	Embedded Controller.
<b>KBC</b>	Keyboard Controller.
<b>Master or SMBus Master</b>	The device that initiates and terminates all communication and drives the clock, SCL.
<b>PEC</b>	Packet error code.
<b>POR</b>	Power on reset.
<b>RTS</b>	Remote temperature sensor, typical examples are ADM1032, LM99, MAX6657, EMC1002.
<b>SBI</b>	Sideband interface.
<b>SB-RMI</b>	Remote Management interface.
<b>Slave or SMBus slave</b>	The slave cannot initiate SMBus communication and cannot drive the clock but can drive the data signal SDA and the alert signal ALERT_L.
<b>TSI</b>	Temperature sensor interface.

### 4.2 SBI Bus Characteristics

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

#### 4.2.1 SMBus Protocol Support

The SBI follows SMBus protocol except:

- The processor does not implement SMBus master functionality.
- The SBI implements the Send Byte/Receive Byte, Read Byte/Write Byte, Block Read/Block Write and Block Write-Block Read Process Call SMBus protocols. The Send Byte/Receive Byte SMBus protocol is only supported by SB-TSI.
- Packet error checking (PEC) is not supported by SB-TSI.
- Address Resolution Protocol (ARP) is not implemented.
- Cumulative clock extensions are not enforced.



## 4.2.2 I2C Support

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

## 4.3 SBI Processor Information

### 4.3.1 SBI Processor Pins

Up to six processor pins are used for SBI support: two for data transfer, three for address determination and one for an interrupt output. Of the three address pins, one bit is socket\_id used to determine which package is addressed. These pins do not have changeable pinstrap. The Serial Interface Clock (SIC) and Serial Interface Data (SID) pins function as the SMBus clock and data pins respectively. The SMBus alert pin (ALERT\_L) is used to signal interrupts to the SMBus master.

#### 4.3.1.1 Physical Layer Characteristics

The SIC and SID pins differ from the SMBus specification with regard to voltage. System board voltage translators are necessary to convert the SIC and SID pin voltage levels to that of the SMBus specification. SBI supports frequencies of 100 KHz, 400 KHz over SIC.

### 4.3.2 Processor States

SBI responds to SMBus 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.

## 4.4 SBI Protocols

### 4.4.1 SBI Modified Block Write-Block Read Process Call

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

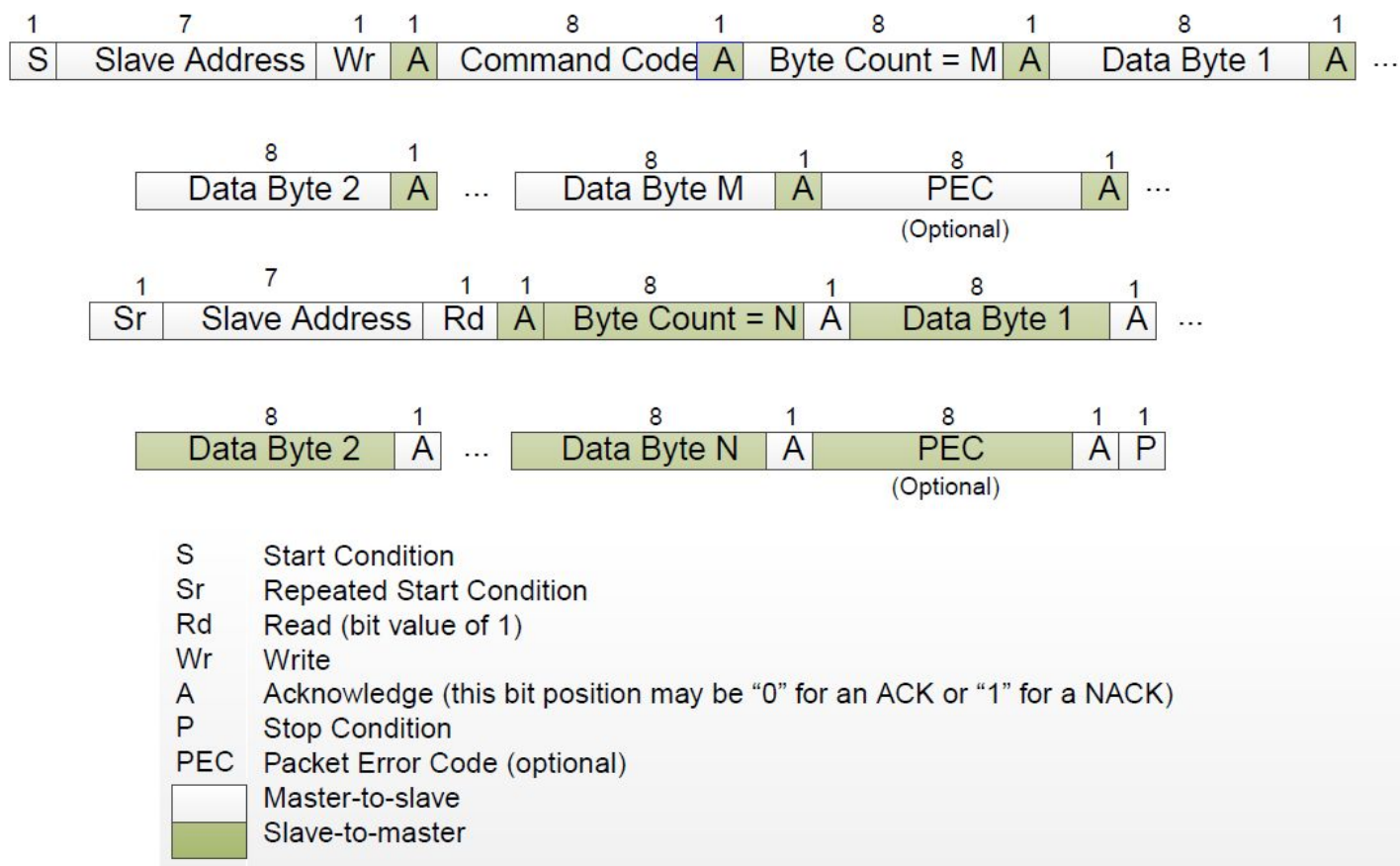


Figure 24: SBI Transmission Protocol

#### 4.4.2 SBI Remote Management Interface (SB-RMI)

SB-RMI provides an interface for an external SMBus master that can be used to perform tasks such as monitoring the processor MCA registers, processor CPUID registers etc. SB-RMI supports signaling Alert\_L when a MCE is received by any thread or when software sets SBRMI::Status[SwAlertSts]. Each package has an independent SMBUS slave port. See 4.5.1 [SBI SMBus Address].

Each package is required to contain the same number of logical threads. The SMBUS slave 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.

##### 4.4.2.1 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 84: SB-RMI Functions

Function	Description	Thread Specific
CPUID	Access to CPUID registers. General purpose registers are not altered unlike a	Y

	processor CUID instruction. Use Read CUID Command Protocol where CUID Function is placed into WrData[7:4] and register is placed in WrData[8]. Access is Read-only.	
MCA Registers	Register read command using the register address to access Core::X86::Msr::MCG_CAP determines the number of MCA banks.  Use Read Processor Register Command Protocol where MSR address is placed into WrData[7:4]. Access is Read-only.	Y
DRAM Throttle	Register read or write command to access the DRAM Controller Command Throttle Register.  The thread number field is not used for this request. Writes are uniformly applied to all DRAM Controller Command Throttle Register Instances within a package. Reads return Dram Controller Command Throttle Register instance 0. Access is Read-write.	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 mailbox specific details.	N
Boot Status	Boot Status is placed in outbound message register SBRMI::MP0OutBndMsg. Access is Read-only.	N

#### 4.4.2.1.1 SB-RMI Read Processor Register and Read CUID Commands

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

*Table 85: SB-RMI Read Processor Register Command Protocol*

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address.
2	Command	73h	Read CUID/Read Register Command Format.
3	WrDataLen	07h	7 Bytes.
4	WrData1	0Xh	Number of bytes to read from register. Valid values are 1 through 8.
5	WrData2	86h	Read Register command.
6	WrData3	XXXX_XXXXb	Bit[0] is Reserved. Bits[7:1] select the thread to address. 00h=Thread0. ... 7Fh=Thread127.
7	WrData4	XXh	Register Address[7:0] from the SB-RMI Functions table.
8	WrData5	XXh	Register Address[15:8] from the SB-RMI Functions table.
9	WrData6	XXh	Register Address[23:16] from the SB-RMI Functions table.
10	WrData7	XXh	Register Address[31:24] from the SB-RMI Functions table.
11	PEC	XXh	Optional PEC byte.
12	Slave Address	0111_XXX1b	Read Address.
13	RdDataLen	0Xh	Number of bytes returned = WrData1+1.

14	Status	XXh	Status Code.
15	RdData1	XXh	Register Data[7:0].
16	RdData2	XXh	Register Data[15:8]. Optional.
17	RdData3	XXh	Register Data[23:16].
18	RdData4	XXh	Register Data[31:24]. Optional.
19	RdData5	XXh	Register Data[39:32]. Optional.
20	RdData6	XXh	Register Data[47:40]. Optional.
21	RdData7	XXh	Register Data[55:48]. Optional.
22	RdData8	XXh	Register Data[63:56]. Optional.
23	PEC	XXh	Optional PEC byte.

Table 86: SB-RMI Read CPUID Command Protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address.
2	Command	73h	Read CPUID/Read Register Command Format.
3	WrDataLen	08h	8 Bytes.
4	WrData1	08h	Number of CPUID bytes to read.
5	WrData2	91h	Read CPUID command.
6	WrData3	XXXX_XXXXb	Bit[0] is Reserved. Bits[7:1] select the thread to address. 00h=Thread0. ... 7Fh=Thread127.
7	WrData4	XXh	CPUID Function[7:0].
8	WrData5	XXh	CPUID Function[15:8].
9	WrData6	XXh	CPUID Function[23:16].
10	WrData7	XXh	CPUID Function[31:24].
11	WrData8	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.
12	PEC	XXh	Optional PEC byte.
13	Slave Address	0111_XXX1b	Read Address.
14	RdDataLen	09h	Number of bytes returned.
15	Status	XXh	Status Code.
16	RdData1	XXh	eax or ecx bits[7:0].
17	RdData2	XXh	eax or ecx bits[15:8].
18	RdData3	XXh	eax or ecx bits[23:16].
19	RdData4	XXh	eax or ecx bits[31:24].
20	RdData5	XXh	ebx or edx bits[7:0].
21	RdData6	XXh	ebx or edx bits[15:8].
22	RdData7	XXh	ebx or edx bits[23:16].
23	RdData8	XXh	ebx or edx bits[31:24].
24	PEC	XXh	Optional PEC byte.

Table 87: SB-RMI Read Data/Status Command Protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address.
2	Command	72h	Read CPUID/Read Register Command Format.

3	WrDataLen	01h	1 byte of Write data.
4	WrData1	0Xh	Number of bytes to read from register. Valid values are 1 through 8.
5	PEC	XXh	Optional PEC byte.
6	Slave Address	0111_XXX1b	Read Address.
7	RdDataLen	0Xh	Number of bytes returned = WrData1 + 1.
8	Status	XXh	Status Code.
9	RdData1	XXh	Register Data[7:0]. Optional.
10	RdData2	XXh	Register Data[15:8]. Optional.
11	RdData3	XXh	Register Data[23:16]. Optional.
12	RdData4	XXh	Register Data[31:24]. Optional.
13	RdData5	XXh	Register Data[39:32]. Optional.
14	RdData6	XXh	Register Data[47:40]. Optional.
15	RdData7	XXh	Register Data[55:48]. Optional.
16	RdData8	XXh	Register Data[63:56]. Optional.
17	PEC	XXh	Optional PEC byte.

#### 4.4.2.1.2 SB-RMI Write Processor Register Command

Writing processor registers from SB-RMI uses two SBI Modified Block Write-Block Read Process Call commands. The first command loads the address of the register to be written into the processor. The register address loaded by this command is stored on a per-thread basis. The second command writes the data to the processor register using the stored register address. The read data/status command can be used to determine that the command completed if a SMBus timeout occurs. The previous command must be complete before a new command can be issued. WrData Address ranges beyond 32 bits are ignored.

Write Register/Load Address command is only used for DRAM throttle feature for address C001\_0079.

*Table 88: SB-RMI Load Address Command Protocol*

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address.
2	Command	71h	Write Register/Load Address Command Format.
3	WrDataLen	06h	6 bytes.
4	WrData1	81h	Load Address Command.
5	WrData2	XXXX_XXXXb	Bit[0] is Reserved. Bits[7:1] select the thread to address. 00h=Thread0. ... 7Fh=Thread127.
6	WrData3	XXh	Register Address[7:0] from SB-RMI Functions table.
7	WrData4	XXh	Register Address[15:8] from SB-RMI Functions table.
8	WrData5	XXh	Register Address[23:16] from SB-RMI Functions table.
9	WrData6	XXh	Register Address[31:24] from SB-RMI Functions table.
10	PEC	XXh	Optional PEC byte.
11	Slave Address	0111_XXX1b	Read Address.
12	RdDataLen	01h	Number of bytes returned.
13	Status	XXh	Status Code.
14	PEC	XXh	Optional PEC byte.

Table 89: SB-RMI Write Processor Register Command Protocol

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address.
2	Command	71h	Write Register/Load Address Command Format.
3	WrDataLen	0Xh	Total number of WrData bytes sent by the master. The total number of bytes written to the register (WrDataLen - 2) must match the size of the register that is being written or undefined data will be written into the register.
4	WrData1	87h	Write Register Command.
5	WrData2	XXXX_XXXXb	Bit[0] is Reserved. Bits[7:1] select the thread to address. 00h=Thread0. ... 7Fh=Thread127.
6	WrData3	XXh	Register Data[7:0].
7	WrData4	XXh	Register Data[15:8]. Optional.
8	WrData5	XXh	Register Data[23:16]. Optional.
9	WrData6	XXh	Register Data[31:24]. Optional.
10	WrData7	XXh	Register Data[39:32]. Optional.
11	WrData8	XXh	Register Data[47:40]. Optional.
12	WrData9	XXh	Register Data[55:48]. Optional.
13	WrData10	XXh	Register Data[63:56]. Optional.
14	PEC	XXh	Optional PEC byte.
7+WrDataLen	Slave Address	0111_XXX1b	Read Address.
8+WrDataLen	RdDataLen	01h	Number of bytes returned.
9+WrDataLen	Status	XXh	Status Code.
10+WrDataLen	PEC	XXh	Optional PEC byte.

#### 4.4.2.1.3 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.

Table 90: SB-RMI Status Codes

Status Code	Name	Description
00h	Success	Command.
11h	Command Timeout	Command did not complete before an SMBus timeout occurred. 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.
42h	Excessive Data Length	The sum of the RdDataLen and WrDataLen is greater than 32 and RdDataLen is greater than or equal to 1 and less than or equal to 32.
44h	Invalid thread	Invalid thread selected.
45h	Unsupported	Command not supported by the processor.

	Command	
81h	Command Aborted	The processor core targeted by the command could not start the command and was aborted by the processor.

#### 4.4.2.2 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 91: SB-RMI Soft Mailbox Message

Command	Message	Description	Command Data In	Command Data Out
31h	WRITE_FAST_PPT_LIMIT	Set APU power limit for system power supply peak control.	[31:0]=fPPT in mW.	None
34h	WRITE_THERM_CTL_LIMIT	Set the thermal throttling limit.	[31:0]=Therm limit in degree Celsius.	None
35h	WRITE_VRM_VDD_CURRENT_LIMIT	Set VDDCR_VDD TDC.	[31:0]=VDD TDC in mA.	None
36h	WRITE_VRM_VDD_MAXIMUM_CURRENT_LIMIT	Set VDDCR_VDD EDC.	[31:0]=VDD EDC in mA.	None

##### 4.4.2.2.1 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 91 [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 92: 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	01h



	be ignored.	
Unknown Command	Unknown mailbox message.	02h
Invalid Core	Invalid core is specified in mailbox message parameters.	03h

The mailbox error code is written by Firmware in SBRMI::OutBndMsg\_inst7 (SBRMI\_x37).

#### 4.4.2.3 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.

#### 4.4.2.4 SB-RMI Register Access

The SB-RMI registers can be read or written from the SMBus interface using the SMBus defined PEC-optional Read Byte and Write Byte protocols with the SB-RMI register number in the command byte or the PEC-optional Block Read and Block Write protocols with the first SB-RMI register number to be accessed in the command byte. 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.

##### 4.4.2.4.1 SB-RMI Register Block Access

The following example shows a write from SBRMI\_x18 to SBRMI\_x1F using SMBus Block Write protocol with SBRMI::Control[BlkRWEn] set to 1.

*Table 93: SB-RMI Register Block Write Protocol*

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address.
2	Command	18h	Indicates starting register SBRMI_x18.
3	Byte Count	08h	Number of bytes to write.
4	Data Byte 1	00h	Write a value to SBRMI_x18h.
5	Data Byte 2	00h	Write a value to SBRMI_x19h.
6	Data Byte 3	00h	Write a value to SBRMI_x1Ah.
7	Data Byte 4	00h	Write a value to SBRMI_x1Bh.
8	Data Byte 5	00h	Write a value to SBRMI_x1Ch.
9	Data Byte 6	00h	Write a value to SBRMI_x1Dh.
10	Data Byte 7	00h	Write a value to SBRMI_x1Eh.
11	Data Byte 8	00h	Write a value to SBRMI_x1Fh.
12	PEC	XXh	Optional PEC byte.

The following example shows a read from SBRMI\_x10 to SBRMI\_x17 using SMBus Block Read protocol with SBRMI::Control[BlkRWEn] set to 1 and SBRMI::ReadSize[RdSize] set to 8.

*Table 94: SB-RMI Register Block Read Protocol*

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address.
2	Command	10h	Indicates starting register SBRMI_x10.
3	Slave Address	0111_XXX1b	Read Address.



4	Byte Count	08h	Number of bytes to read.
5	Data Byte 1	00h	Read a value from SBRMI_x10h.
6	Data Byte 2	00h	Read a value from SBRMI_x11h.
7	Data Byte 3	00h	Read a value from SBRMI_x12h.
8	Data Byte 4	00h	Read a value from SBRMI_x13h.
9	Data Byte 5	00h	Read a value from SBRMI_x14h.
10	Data Byte 6	00h	Read a value from SBRMI_x15h.
11	Data Byte 7	00h	Read a value from SBRMI_x16h.
12	Data Byte 8	00h	Read a value from SBRMI_x17h.
13	PEC	XXh	Optional PEC byte.

#### 4.4.2.4.2 SB-RMI Register Byte Access

The following example shows a write to SBRMI\_x03 using the SMBus Write Byte protocol with SBRMI::Control[BlkRWEn] set to 0.

*Table 95: SB-RMI Register Write Byte Protocol*

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

The following example shows a read from SBRMI\_x03 using the SMBus Read Byte protocol with SBRMI::Control[BlkRWEn] set to 0.

*Table 96: SB-RMI Register Read Byte Protocol*

Byte	Byte Name	Value	Notes
1	Slave Address	0111_XXX0b	Write Address.
2	Command	03h	Indicates SB-RMI register 03.
3	Slave Address	0111_XXX1b	Read address.
4	Data Byte	04h	Value of SBRMI_x03h.
5	PEC	XXh	Optional PEC byte.

#### 4.4.2.5 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].

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.

### 4.4.3 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 master and slave devices to disagree on when data should be stable and when it is allowed to be changing.

#### **4.4.3.1 Error Detection**

SBI provides several methods of error detection: protocol ACK/NAK, packet error correction (PEC) fields, and timeouts. The ACK/NAK mechanism is always active in SBI, but the PEC and timeouts are optional.

##### **4.4.3.1.1 ACK/NAK Mechanism**

After each byte of an SBI message, the device receiving that byte must either acknowledge (ACK) that it received the byte correctly, or deny (NAK) that the byte was correctly received. This is most easily seen in the case of the address bytes which follow a START (or REPEATED START) sequence, but can be used anywhere in the message. In the case of an address byte, if a slave device recognizes the address, it will respond with an ACK and await the rest of the message. If a slave device does not recognize the message, it will respond with a NAK and ignore the rest of the message.

##### **4.4.3.1.2 Packet Error Correction (PEC)**

The RMI protocols allow for PEC bytes to be appended to messages. The sending side calculates the PEC, based on the data it intends to transmit, and appends it to the transmitted data. The receiving side calculates the PEC based on the data it actually receives and compares that to the PEC it receives. If the two PECs do not match, an error has occurred and the message should be discarded. When a device detects a PEC mismatch, it should send a NAK in response to the PEC. No special programming is needed to enable the PEC on AMD devices. If the PEC is present on an incoming message, the device will verify the PEC and ACK or NAK as appropriate. The PEC is always calculated on outgoing messages. It is up to the bus master to request the PEC by sending clocks for that byte before sending either a NAK or a STOP sequence.

##### **4.4.3.1.3 Bus Timeouts**

Bus timeouts should be enabled to prevent a device waiting indefinitely on a message that may not be coming. Some timeouts are used to prevent the SBI bus from waiting for a response from a CPU that is in a power-saving idle mode. Other timeouts are used to allow the slave device to recognize that the bus master is attempting to reset all of the devices on the SBI bus. Either way, when a device recognizes a timeout, it should abort its current message transfer.

#### **4.4.3.2 Error Recovery**

The simplest form of error recovery is a retry. When the bus master detects an unexpected NAK, 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 slave device to be selected. That slave 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.

##### **4.4.3.2.1 SBI Bus Reset**

The bus master can hold the clock low for a period longer the standard timeout in order to force slave devices off the bus (see docSMB section 3.1.1.3 of the System Management Bus (SMBus) Specification, version 2.0). All SBI slave devices are required to reset their communications if another device holds the clock line low for longer than TTimeout, min (25 milliseconds). The devices are required to complete their reset within TTimeout, max (35 milliseconds). SBI bus masters should use the extended timeout to force a reset of all slave devices if a simple retry does not remove an error condition.

## 4.5 SBI Physical Interface

### 4.5.1 SBI SMBus Address

The SMBus address is really 7 bits. Some vendors and the SMBus specification show the address as 8 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. Some vendors use only the 7 bits to describe the address.

### 4.5.2 SBI Bus Timing

SBI supports 100KHz standard-mode and 400 KHz fast-mode I2C operation. Refer to the standard-mode and fast-mode timing parameters in the I2C specification.

## 4.6 SB-RMI Registers

Reads to unimplemented registers may return non zero value. Writes to unimplemented registers are discarded.

### SBRMIx00 [Revision] (SBRMI::Revision)

Read-only. Reset: 10h.

Bits	Description
7:0	<b>Revision: SB-RMI revision.</b> Read-only. Reset: 10h. This field specifies the APLM specification revision that the product is compliant to. 0x10=1.0x Revision.

### SBRMIx01 [Control] (SBRMI::Control)

Read-write. Reset: 01h.

Bits	Description
7	<b>PECEn: packet error checking enable.</b> Read-write. Reset: 0. This only controls the intermediate PEC of the SBI Modified Block Write-Block Read Process Call. 0=Intermediate PEC is disabled. 1=Intermediate PEC is enabled.
6:5	Reserved.
4	<b>SwAlertMask: software alert mask.</b> Read-write. Reset: 0. 0=Alert_L signaling is enabled when SBRMI_x02SwAlertSts is set. 1=Alert_L signaling is disabled when SBRMI_x02SwAlertSts is set.
3	<b>BlkRWEn: block read/write enable.</b> Read-write. Reset: 0. Controls Block Read/Write access to register ranges SBRMI_x[4F:10] and SBRMI_x[9F:80]. 0=SMBus accesses can only use the Byte Read/Write protocol. 1=SMBus accesses can only use the Block Read/Write protocol. NOTE: All other register ranges only support Byte Read/Write access, independent of the state of the BlkRWEn control bit.
2	<b>TimeoutDis: SB-RMI timeout disable.</b> Read-write. Reset: 0. 1=SMBus defined timeouts are disabled. If the SB-TSI interface is also in use, SMBus timeouts should be enabled or disabled in a consistent manner on both interfaces. The SB-TSI timeout setting is used by SB-RMI until the SMBus interface can determine which interface is targeted by the transaction.
1	<b>AraDis: SB-RMI ARA disable.</b> Read-write. Reset: 0. 1=Sending of an ARA response is disabled. 0=Sending of an ARA response is enabled.
0	<b>AlertMask: SB-RMI alert mask.</b> Read-write. Reset: 1. Read-write; set-by-hardware if AraDis=0 and a successful ARA is sent. 1=Alert_L signaling disabled. 0=Alert_L is asserted if any unmasked event is present in the [The Alert Status Registers] SBRMI_x1[F:0], or if SBRMI_x02[SwAlertSts] == 1 and SwAlertMask == 0.

**SBRMIx02 [Status] (SBRMI::Status)**

Reset: 00h.

Bits	Description
7:2	Reserved.
1	<b>SwAlertSts: SB-RMI software alert status.</b> Read-write, Volatile. Reset: 0. Write-one-to-clear from the SMBus interface; Read-write from the processor. Set by firmware as a result of a Machine Check Exception prior to the MCE related warm reset. Set by firmware to indicate the completion of a mailbox operation.
0	<b>AlertSts: SB-RMI alert status.</b> Read-only, Volatile. Reset: 0. Read-only. 1=Alert event present in SBRMI::AlertStatus.

**SBRMIx03 [Read Size] (SBRMI::ReadSize)**

Read-write. Reset: 01h.

This register specifies the number of bytes to return when using the block read protocol to read SBRMI\_x[4F:10] and SBRMI\_x[90:80].

Bits	Description								
7:6	Reserved.								
5:0	<b>RdSize: read size.</b> Read-write. Reset: 01h. Specifies the number of bytes to return when using the block read protocol.								
<b>Valid Values:</b>									
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>00h</td><td>Reserved.</td></tr> <tr> <td>20h-01h</td><td>&lt;Value&gt; 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.								

**SBRMIx0[4...9] [Thread Enable Status] (SBRMI::ThreadEnableStatus)**

Read-only.

\_inst[3:0]; SBRMIx[09,08,05,04]

Bits	Description		
7:0	<b>threadEnStat: thread enable status.</b> Read-only.		
	<b>Description:</b> 1=Thread is enabled.		
	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].	

**SBRMIx1[0...F] [Alert Status] (SBRMI::AlertStatus)**

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

\_inst[15:0]; SBRMIx1[F:0]

Bits	Description	
7:4	Reserved.	
3:0	<b>MceStat: MCE status.</b> Read,Write-1-to-clear,Volatile.	
	<b>Description:</b> Bit vector for threads. 1=MCE occurred for thread. Set by hardware.	
	Offset[7:0]	inst Description
	10h	0 Threads[48,32,16,0].
	11h	1 Threads[49,33,17,1].
	12h	2 Threads[50,34,18,2].
	13h	3 Threads[51,35,19,3].
	14h	4 Threads[52,36,20,4].
	15h	5 Threads[53,37,21,5].
	16h	6 Threads[54,38,22,6].
	17h	7 Threads[55,39,23,7].
	18h	8 Threads[56,40,24,8].
	19h	9 Threads[57,41,25,9].
	1Ah	10 Threads[58,42,26,10].
	1Bh	11 Threads[59,43,27,11].
	1Ch	12 Threads[60,44,28,12].
	1Dh	13 Threads[61,45,29,13].
	1Eh	14 Threads[62,46,30,14].
	1Fh	15 Threads[63,47,31,15].

**SBRMIx2[0...F] [Alert Mask] (SBRMI::AlertMask)**

Read-write.

\_inst[15:0]; SBRMIx2[F:0]

Bits	Description																																																			
7:4	Reserved.																																																			
3:0	<b>MceAlertMsk: MCE alert mask.</b> Read-write. <b>Description:</b> Bit vector for threads. 1=Alert signaling disabled for corresponding SBRMI::AlertStatus[MceStat] for thread. <table><tr><th>Offset[7:0]</th><th>inst</th><th>Description</th></tr><tr><td>20h</td><td>0</td><td>Threads[48,32,16,0].</td></tr><tr><td>21h</td><td>1</td><td>Threads[49,33,17,1].</td></tr><tr><td>22h</td><td>2</td><td>Threads[50,34,18,2].</td></tr><tr><td>23h</td><td>3</td><td>Threads[51,35,19,3].</td></tr><tr><td>24h</td><td>4</td><td>Threads[52,36,20,4].</td></tr><tr><td>25h</td><td>5</td><td>Threads[53,37,21,5].</td></tr><tr><td>26h</td><td>6</td><td>Threads[54,38,22,6].</td></tr><tr><td>27h</td><td>7</td><td>Threads[55,39,23,7].</td></tr><tr><td>28h</td><td>8</td><td>Threads[56,40,24,8].</td></tr><tr><td>29h</td><td>9</td><td>Threads[57,41,25,9].</td></tr><tr><td>2Ah</td><td>10</td><td>Threads[58,42,26,10].</td></tr><tr><td>2Bh</td><td>11</td><td>Threads[59,43,27,11].</td></tr><tr><td>2Ch</td><td>12</td><td>Threads[60,44,28,12].</td></tr><tr><td>2Dh</td><td>13</td><td>Threads[61,45,29,13].</td></tr><tr><td>2Eh</td><td>14</td><td>Threads[62,46,30,14].</td></tr><tr><td>2Fh</td><td>15</td><td>Threads[63,47,31,15].</td></tr></table>	Offset[7:0]	inst	Description	20h	0	Threads[48,32,16,0].	21h	1	Threads[49,33,17,1].	22h	2	Threads[50,34,18,2].	23h	3	Threads[51,35,19,3].	24h	4	Threads[52,36,20,4].	25h	5	Threads[53,37,21,5].	26h	6	Threads[54,38,22,6].	27h	7	Threads[55,39,23,7].	28h	8	Threads[56,40,24,8].	29h	9	Threads[57,41,25,9].	2Ah	10	Threads[58,42,26,10].	2Bh	11	Threads[59,43,27,11].	2Ch	12	Threads[60,44,28,12].	2Dh	13	Threads[61,45,29,13].	2Eh	14	Threads[62,46,30,14].	2Fh	15	Threads[63,47,31,15].
Offset[7:0]	inst	Description																																																		
20h	0	Threads[48,32,16,0].																																																		
21h	1	Threads[49,33,17,1].																																																		
22h	2	Threads[50,34,18,2].																																																		
23h	3	Threads[51,35,19,3].																																																		
24h	4	Threads[52,36,20,4].																																																		
25h	5	Threads[53,37,21,5].																																																		
26h	6	Threads[54,38,22,6].																																																		
27h	7	Threads[55,39,23,7].																																																		
28h	8	Threads[56,40,24,8].																																																		
29h	9	Threads[57,41,25,9].																																																		
2Ah	10	Threads[58,42,26,10].																																																		
2Bh	11	Threads[59,43,27,11].																																																		
2Ch	12	Threads[60,44,28,12].																																																		
2Dh	13	Threads[61,45,29,13].																																																		
2Eh	14	Threads[62,46,30,14].																																																		
2Fh	15	Threads[63,47,31,15].																																																		

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

Read-write. Reset: 00h.

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

Bits	Description																										
7:0	<b>OutBndMsg: outbound message data.</b> Read-write. Reset: 00h.																										
	<b>Description:</b> Read-write from the processor; Read-only from the SMBus interface.																										
	Usage convention is:																										
	<ul style="list-style-type: none"><li>SBRMI::OutBndMsg_inst0 is command copied by firmware from SBRMI::InBndMsg_inst0.</li><li>SBRMI::OutBndMsg_inst[4:1] are 32-bit data.</li><li>SBRMI::OutBndMsg_inst[6:5] are Reserved.</li><li>SBRMI::OutBndMsg_inst[7] contains Mailbox Error Code, per Table 92 [SB-RMI Soft Mailbox Error Code]</li></ul>																										
	<table><tr><th>Offset[7:0]</th><th>inst</th><th>Description</th></tr><tr><td>30h</td><td>0</td><td>Outbound message 0.</td></tr><tr><td>31h</td><td>1</td><td>Outbound message 1.</td></tr><tr><td>32h</td><td>2</td><td>Outbound message 2.</td></tr><tr><td>33h</td><td>3</td><td>Outbound message 3.</td></tr><tr><td>34h</td><td>4</td><td>Outbound message 4.</td></tr><tr><td>35h</td><td>5</td><td>Outbound message 5.</td></tr><tr><td>36h</td><td>6</td><td>Outbound message 6.</td></tr><tr><td>37h</td><td>7</td><td>Outbound message 7.</td></tr></table>	Offset[7:0]	inst	Description	30h	0	Outbound message 0.	31h	1	Outbound message 1.	32h	2	Outbound message 2.	33h	3	Outbound message 3.	34h	4	Outbound message 4.	35h	5	Outbound message 5.	36h	6	Outbound message 6.	37h	7
Offset[7:0]	inst	Description																									
30h	0	Outbound message 0.																									
31h	1	Outbound message 1.																									
32h	2	Outbound message 2.																									
33h	3	Outbound message 3.																									
34h	4	Outbound message 4.																									
35h	5	Outbound message 5.																									
36h	6	Outbound message 6.																									
37h	7	Outbound message 7.																									

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

Read-write. Reset: 00h.

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

Bits	Description		
7:0	<b>InBndMsg: inbound message data.</b> Read-write. Reset: 00h.		
	<b>Description:</b> Read-write from the SMBus 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"><li>SBRMI::InBndMsg_inst0 is command.</li><li>SBRMI::InBndMsg_inst[4:1] are 32-bit data.</li><li>SBRMI::InBndMsg_inst[6:5] are Reserved.</li><li>SBRMI::InBndMsg_inst7: Bit[7] Must be 1'b1 to send message to firmware.</li></ul>		
	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 SMBus master to generate an interrupt to the processor to indicate that a message is available.

Bits	Description
7:1	Reserved.
0	<b>SwInt: firmware interrupt.</b> Read,Write-1-only. Reset: 0. Read,Write-1-only from the SMBus interface; Read,Write-1-to-clear from firmware. 1=Indicates a firmware mailbox service request.

**SBRMIx41 [Thread Number] (SBRMI::ThreadNumber)**

Read-write. Reset: 00h.

This register indicates the maximum number of threads present.

Bits	Description
7:0	<b>threadNum: thread number.</b> Read-write. Reset: 00h. Read-only from the SMBus interface. Specifies the maximum number of threads present. Format is [6:1] number of threads – 40h and range of available threads 40h – 01h. 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] [MP0 Out-Bound Message] (SBRMI::MP0OutBndMsg)		
Read-write. Reset: 00h.		
_inst[7:0]; SBRMIx8[7:0]		
Bits	Description	
7:0	<b>MP0OutBndMsg: outbound message data.</b> Read-write. Reset: 00h.	
	<b>Description:</b> Read-write from the processor; Read-only from the SMBus interface.	
	These registers are used for sending messages from PSP firmware running on the MP0 to the SMBus master.	
	MP0 boot status is dynamically written to this register during the boot process.	
	Offset[7:0]	inst    Description
	80h	0    MP0 Outbound message 0.
	81h	1    MP0 Outbound message 1.
	82h	2    MP0 Outbound message 2.
	83h	3    MP0 Outbound message 3.
	84h	4    MP0 Outbound message 4.
	85h	5    MP0 Outbound message 5.
	86h	6    MP0 Outbound message 6.
	87h	7    MP0 Outbound message 7.



## 5 SB Temperature Sensor Interface (SB-TSI)

### 5.1 Overview

The SBI temperature sensor interface (SB-TSI) is an emulation of the software and physical interface of a typical 8-pin remote temperature sensor (RTS), see Figure 25 [RTS Thermal Management Example]. The goal is to resemble a typical RTS so that KBC or BMC firmware requires minimal changes for future AMD products, see Figure 26 [SB-TSI Thermal Management Example]. SB-TSI supports the SMBus protocols that typical RTS supports.

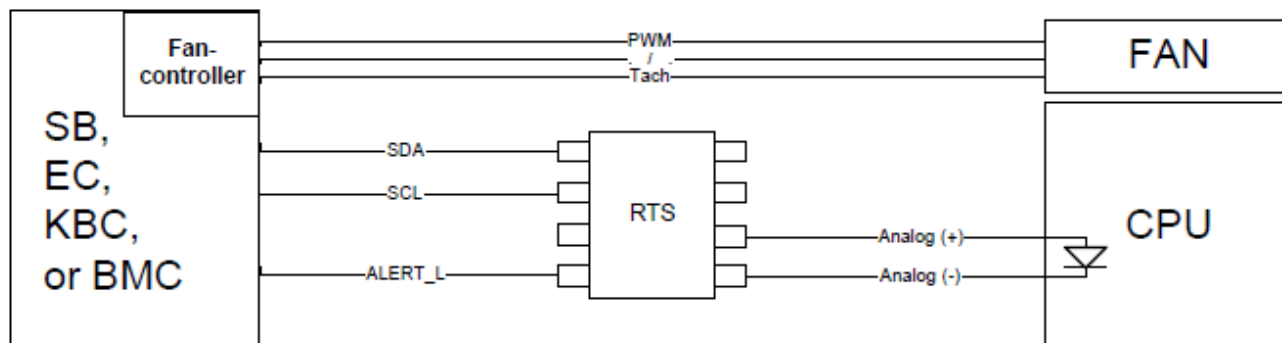


Figure 25: RTS Thermal Management Example

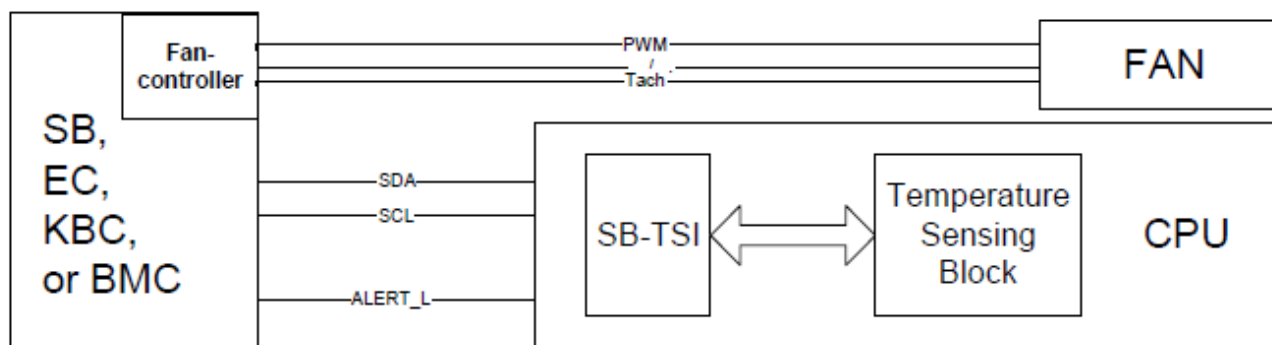


Figure 26: SB-TSI Thermal Management Example

Refer to the following external sources for additional information.

- System Management Bus (SMBus) specification. See docSMB.
- I2C-bus Specification and User Manual, Revision 03. See docI2C.

### 5.1.1 Definitions

Table 97: SB-TSI Definitions

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

## 5.2 SB-TSI Protocol

The SB-TSI largely follows SMBus v2.0 specification except:

- The combined-format repeated start sequence is not supported in standard-mode and fast-mode. The response of the processor's SB-TSI to the sequence is undefined.
- Only 7-bit SMBus addresses are supported.
- SB-TSI implements the Send/Receive Byte and Read/Write Byte protocols.
- SB-TSI registers can only be written by using a Write byte command.
- Address Resolution Protocol (ARP) is not supported.
- Packet Error Checking (PEC) is not supported.
- The usage of unsupported protocols may lead to an undefined bus condition.
- To release the bus from an undefined condition and to reset the SB-TSI slave, the bus master must hold the clock low for a duration of time that is longer than Ttimeout.max, as specified for SMBus. The time-out needs to be enabled by SBTsi::TimeoutConfig[TimeoutEn] = 1.

### 5.2.1 SB-TSI Send/Receive Byte Protocol

A SMBus master can Read SB-TSI registers by issuing a send byte command with the address of the register to be read as the data byte followed by a receive byte command.

#### 5.2.1.1 SB-TSI Address Pointer

The SB-TSI controller has an internal address pointer that is updated when a register is accessed using a Read or Write byte command or when a send byte command is received. This address pointer is used to determine the address of the register being read when a receive byte command is processed by the controller.

### 5.2.2 SB-TSI Read/Write Byte Protocol

An SMBus master can Read or Write SB-TSI registers by issuing a Read or a Write byte command with the address of the register to be read or written in the command code field.

### 5.2.3 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::Config[AraDis]: Disables ARA response.
- 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. Affects ARA.

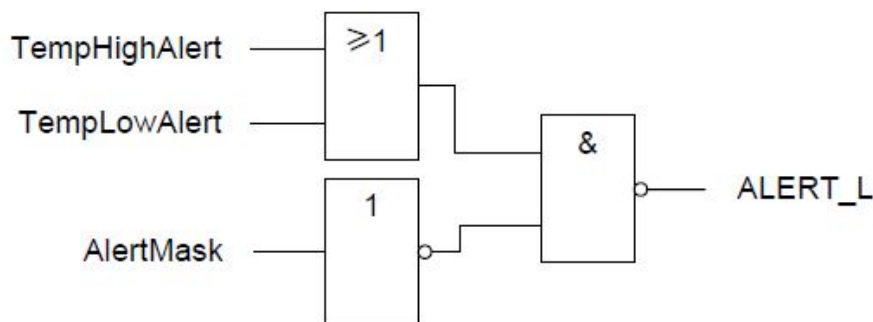


Figure 27: Alert Assertion Diagram

#### 5.2.4 Atomic Read Mechanism

To ensure that the two required Reads (integer and decimal) for reading the CPU temperature are always originated from one temperature value, atomic reading procedures are required. SB-TSI offers functions to maintain atomicity between the temperature integer and decimal bytes.

SBTSI::Config[ReadOrder] specifies the order for reading below 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 latch syncs the decimal part with the integer part. The integer part is continuously updated.

If SBTSI::Config[ReadOrder] is 1, then the Read order to ensure atomicity is Reversed, i.e., decimal part = first, integer part = second.

If it is not possible to ensure a dedicated Read order as described above, the Run/Stop bit ([The SB-TSI Configuration Register] 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 it is 1, they get frozen and always deliver their last value on Read requests.

- 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].

#### 5.2.5 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 98: 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

## 5.2.6 SB-TSI Temperature Offset Encoding

By default, SBTISI::CpuTempInt and SBTISI::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. SBTISI::CpuTempOffInt[CpuTempOffInt] and SBTISI::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 99: 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

## 5.3 SB-TSI Physical Interface

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

### 5.3.1 SB-TSI SMBus Address

The SMBus address is really 7 bits. Some vendors and the SMBus specification show the address as 8 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. Some vendors use only the 7 bits to describe the address. The addresses can vary with address select pins.

*Table 100: SB-TSI Address Encodings*

Socket ID	SB-TSI Address
0b	98h for 8-bit or 4Ch for 7-bit.
1b	90h for 8-bit or 48h for 7-bit.

### 5.3.2 SB-TSI Bus Timing

SB-TSI supports standard-mode (100 kHz) and fast-mode (400 kHz) according to the I2C-bus Specification and User Manual.

### 5.3.3 SB-TSI Bus Electrical Parameters

SB-TSI conforms to most of the I2C fast-mode electrical parameters. See the Electrical Data Sheet for the processor family for electrical parameters.

### 5.3.4 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, as shown in the following diagram.

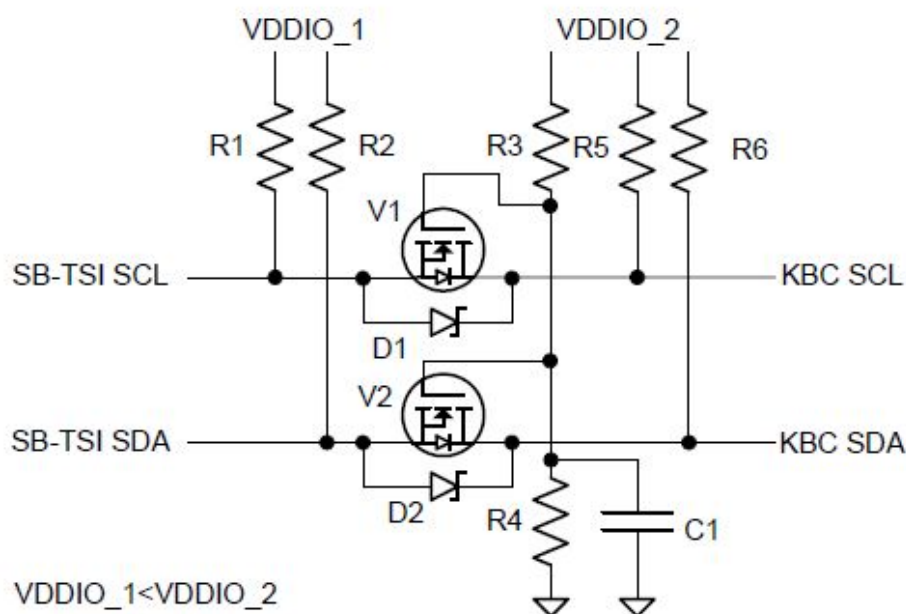


Figure 28: Pass FET Implementation

#### 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).

### 5.4 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 5.2.5 [SB-TSI Temperature and Threshold Encodings]

Bits	Description
7:0	<b>CpuTempInt: integer CPU temperature value.</b> 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 5.2.3 [Alert Behavior].

Bits	Description
7:5	Reserved.
4	<b>TempHighAlert: temperature high alert.</b> 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	<b>TempLowAlert: temperature low alert.</b> 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 5.2.3 [Alert Behavior] and 5.2.4 [Atomic Read Mechanism].

Bits	Description
7	<b>AlertMask: alert mask.</b> Read-only, Volatile. Reset: Cold,0. 0=ALERT_L pin enabled. 1=ALERT_L pin disabled and does not assert. IF (SBTSI::Config[AraDis] == 0) THEN Read-only; set-by-hardware. ELSE Read-only ENDIF. Hardware sets this bit if SBTSI::Config[AraDis] == 0, either SBTSI::Status[TempHighAlert] == 1 or SBTSI::Status[TempLowAlert] == 1, and a successful ARA is sent.
6	<b>RunStop: run stop.</b> 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 5.2.4 [Atomic Read Mechanism] for further details.
5	<b>ReadOrder: atomic read order.</b> 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 5.2.4 [Atomic Read Mechanism] for further details.
4:2	Reserved.
1	<b>AraDis: ARA disable.</b> Read-only. Reset: Cold,0. Read-only. 1=ARA response disabled.
0	Reserved.

**SBTSIx04 [Update Rate] (SBTSI::UpdateRate)**

Read-write. Reset: Cold,08h.

Bits	Description
7:0	<b>UpRate: update rate.</b> 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]).
<b>ValidValues:</b>	
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 5.2.5 [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 5.2.3 [Alert Behavior].

Bits	Description
7:0	<b>HiTempInt: high temperature integer threshold.</b> 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 5.2.5 [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 5.2.3 [Alert Behavior].

Bits	Description
7:0	<b>LoTempInt: low temperature integer threshold.</b> 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	<b>AlertMask: alert mask.</b> Read-write. Reset: Cold,0. See SBTSI::Config[AlertMask].
6	<b>RunStop: run stop.</b> Read-write. Reset: Cold,0. See SBTSI::Config[RunStop].
5	<b>ReadOrder: atomic read order.</b> Read-write. Reset: Cold,0. See SBTSI::Config[ReadOrder].
4:2	Reserved.
1	<b>AraDis: ARA disable.</b> Read-write. Reset: Cold,0. See SBTSI::Config[AraDis].
0	Reserved.

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

Read-only.

See SBTSI::CpuTempInt.

Bits	Description
7:5	<b>CpuTempDec: decimal CPU temperature value.</b> 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 5.2.6 [SB-TSI Temperature Offset Encoding] for encoding details.

Bits	Description
7:0	<b>CpuTempOffInt: CPU temperature integer offset.</b> 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	<b>CpuTempOffDec: CPU temperature decimal offset.</b> 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	<b>HiTempDec: high temperature decimal threshold.</b> 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	<b>LoTempDec: low temperature decimal threshold.</b> Read-write. Reset: Cold,0h. This field specifies the decimal portion of the low temperature threshold.
4:0	Reserved.

**SBTSIx22 [Timeout Configuration] (SBTSI::TimeoutConfig)**

Read-write. Reset: Cold,80h.

Bits	Description
7	<b>TimeoutEn: SMBus timeout enable.</b> Read-write. Reset: Cold,1. 0=SMBus defined timeout support disabled. 1=SMBus defined timeout support enabled. SMBus timeout enable.
6:0	Reserved.

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

Read-write. Reset: Cold,00h.

See 5.2.3 [Alert Behavior].

Bits	Description
7:3	Reserved.
2:0	<b>AlertThr: alert threshold.</b> 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.
<b>ValidValues:</b>	
<b>Value</b>	<b>Description</b>
0h	1 Sample
6h-1h	<Value+1> Samples
7h	8 Samples

**SBTSIxBF [Alert Configuration] (SBTSI::AlertConfig)**

Read-write.

Bits	Description
7:1	Reserved.
0	<b>AlertCompEn: alert comparator mode enable.</b> 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; ARA response disabled. 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	<b>ManId: Manufacture ID.</b> Read-only. Reset: Cold,0. Returns the AMD manufacture ID.

**SBTSIxFF [Revision] (SBTSI::Revision)**

Read-only. Reset: Cold,04h.

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

## List of Namespaces

Namespace	Heading(s)
Core::X86::Apic	2.1.11.2.2 [Local APIC Registers]
Core::X86::Cpuid	2.1.12.1 [CPUID Instruction Functions]
Core::X86::Msr	2.1.13.1 [MSRs - MSR0000_xxxx] 2.1.13.2 [MSRs - MSRC000_xxxx] 2.1.13.3 [MSRs - MSRC001_0xxx] 2.1.13.4 [MSRs - MSRC001_1xxx]
Core::X86::Pmc::Core	2.1.14.4 [Large Increment per Cycle Events] 2.1.14.5.1 [Floating-Point (FP) Events] 2.1.14.5.2 [Load/Store (LS) Events] 2.1.14.5.3 [Instruction Cache (IC) and Branch Prediction (BP) Events] 2.1.14.5.4 [DE Events] 2.1.14.5.5 [EX (SC) Events]
Core::X86::Pmc::L2	2.1.14.5.6 [L2 Cache Events]
Core::X86::Pmc::L3	2.1.14.6.1 [L3 Cache PMC Events]
Core::X86::Smm	2.1.11.1.6 [System Management State]
IO	2.1.7 [PCI Configuration Legacy Access]
L3::L3CRB	2.2.1 [L3 MSR Registers]
L3::L3CT	2.2.2 [L3 Clocks and Test (CT) MSR Registers.]
MCA::CS	3.2.6.8 [CS]
MCA::DE	3.2.6.4 [DE]
MCA::EX	3.2.6.5 [EX]
MCA::FP	3.2.6.6 [FP]
MCA::IF	3.2.6.2 [IF]
MCA::KPX::GMI	3.2.6.14 [KPX GMI]
MCA::KPX::SERDES	3.2.6.13 [KPX SERDES]
MCA::L2	3.2.6.3 [L2]
MCA::L3	3.2.6.7 [L3]
MCA::LS	3.2.6.1 [LS]
MCA::MP5	3.2.6.11 [MP5]
MCA::NBIO	3.2.6.12 [NBIO]
MCA::PCS::GMI	3.2.6.15 [PCS GMI]
MCA::PIE	3.2.6.9 [PIE]
MCA::UMC	3.2.6.10 [UMC]
SBRMI	4.6 [SB-RMI Registers]

SBTSI	5.4 [SB-TSI Registers]
-------	------------------------

## List of Definitions

**ABS**: ABS(integer expression): Remove sign from signed value.

**AGESA™**: AMD Generic Encapsulated Software Architecture.

**AM5**: Desktop, single socket. For client desktop platform (LGA) with DDR5. AM5 = (Core::X86::CpuId::BrandId[PkgType] == 00h).

**AP**: Applications Processor.

**APML**: Advanced Platform Management Link.

**ARA**: Alert response address.

**ARP**: Address Resolution Protocol

**BAR**: The BAR, or base address register, physical register mnemonic format is of the form PREFIXxZZZ. PREFIX is an all capital letter name that connotes the BAR to which the offset is added to get the physical address of the operation. ZZZ is the offset.

**BCD**: Binary Coded Decimal number format.

**BCS**: Base Configuration Space.

**BIST**: Built-In Self-Test. Hardware within the processor that generates test patterns and verifies that they are stored correctly (in the case of memories) or received without error (in the case of links).

**BMC**: Base management controller.

**Boot VID**: Boot Voltage ID. This is the VDD and VDDNB voltage level that the processor requests from the external voltage regulator during the initial phase of the cold boot sequence.

**BSC**: Boot strap core. Core 0 of the BSP.

**BSP**: Boot strap processor.

**C-states**: These are ACPI defined core power states. C0 is operational. All other C-states are low-power states in which the processor is not executing code. See docACPI.

**Canonical-address**: An address in which the state of the most-significant implemented bit is duplicated in all the remaining higher-order bits, up to bit[63].

**CCX**: Core Complex where more than one core shares L3 resources.

**CEIL**: CEIL(real expression): Rounds real number up to nearest integer.

**CMP**: Specifies the core number.

**COF**: Current operating frequency of a given clock domain.

**Cold reset**: PWROK is de-asserted and RESET\_L is asserted.

**Configurable**: Indicates that the access type is configurable as described by the documentation.

**CoreCOF**: Core current operating frequency in MHz. CoreCOF = Core::X86::Msr::PStateDef[CpuFid[11:0]] \* 5MHz.

**COUNT**: COUNT(integer expression): Returns the number of binary 1's in the integer.

**CpuCoreNum**: Specifies the core number.

**CPUID**: The CPUID, or x86 processor identification state, physical register mnemonic format is of the form CPUID FnXXXX\_XXXX\_EiX[\_xYYY], where XXXX\_XXXX is the hex value in the EAX and YYY is the hex value in ECX.

**DID**: Divisor Identifier. Specifies the post-PLL divisor used to reduce the COF.

**docACPI**: Advanced Configuration and Power Interface (ACPI) Specification. <http://uefi.org/specifications>.

**docAPM1**: AMD64 Architecture Programmer's Manual Volume 1: Application Programming, order# 24592.

**docAPM2**: AMD64 Architecture Programmer's Manual Volume 2: System Programming, order# 24593.

**docAPM3**: AMD64 Architecture Programmer's Manual Volume 3: Instruction-Set Reference, order# 24594.

**docAPM4**: AMD64 Architecture Programmer's Manual Volume 4: 128-Bit and 256-Bit Media Instructions, order# 26568.

**docAPM5**: AMD64 Architecture Programmer's Manual Volume 5: 64-Bit Media and x87 Floating-Point Instructions, order# 26569.

**docASF**: Alert Standard Format Specification. <http://dmtf.org/standards/asf>.

**docATA**: AT Attachment with Packet Interface. <http://www.t13.org>.

**docDP**: VESA DisplayPort Standard. <http://www.vesa.org/vesa-standards>.

**docI2C**: I2C Bus Specification. [http://www.nxp.com/documents/user\\_manual/UM10204.pdf](http://www.nxp.com/documents/user_manual/UM10204.pdf)

**docIOMMU**: AMD I/O Virtualization Technology Specification, order# 48882.

**docJEDEC**: JEDEC Standards. <http://www.jedec.org>.

**docPCIe**: PCI Express® Specification. <http://www.pcisig.org>.

**docPCIb**: PCI Local Bus Specification. <http://www.pcisig.org>.

**docSATA**: Serial ATA Specification. <http://www.sata-io.org>.

**docSDHC**: Secure Digital Host Controller Standard Specification.

<https://www.sdcard.org>.

**docUSB**: Universal Serial Bus Specification. <http://www.usb.org>.

**Doubleword**: A 32-bit value.

**DW**: Doubleword.

**EC**: Embedded Controller.

**ECS**: Extended Configuration Space.

**Error-on-read**: Error occurs on read.

**Error-on-write**: Error occurs on write.

**Error-on-write-0**: Error occurs on bitwise write of 0.

**Error-on-write-1**: Error occurs on bitwise write of 1.

**FCH**: The integrated platform subsystem that contains the IO interfaces and bridges them to the system BIOS. Previously included in the Southbridge.

**FID**: Frequency Identifier. Specifies the PLL frequency multiplier for a given clock domain.

**FL1**: Notebook and mobile workstations. FL1 =

(Core::X86::CpuId::BrandId[PkgType] == 01h).

**FLOOR**: FLOOR(integer expression): Rounds real number down to nearest integer.

**GT/s**: Giga-Transfers per second.

**HWPF**: Hardware Prefetcher.

**IBS**: Instruction based sampling.

**IFCM**: Isochronous flow-control mode, as defined in the link specification.

**Inaccessible**: Not readable or writable (e.g., Hide ? Inaccessible : Read-Write).

**IO configuration**: Access to configuration space through IO ports CF8h and CFCh.

**IORR**: IO range register.

**KBC**: Keyboard Controller.

**L1 cache**: The level 1 caches (instruction cache and the data cache).

**L2 cache**: The level 2 caches.

**Linear (virtual) address**: The address generated by a core after the segment is applied.

**LINT**: Local interrupt.

**Logical address**: The address generated by a core before the segment is applied.

**logical mnemonic**: The register mnemonic format that describes the register functionally, what namespace to which the register belongs, a name for the register that connotes its function, and optionally, named parameters that indicate the different function of each instance (e.g., Link::Phy::PciDevVendIDF3). See 1.4.3.1 [Logical Mnemonic].

**LRU**: Least recently used.

**LVT**: Local vector table. A collection of APIC registers that define interrupts for local events (e.g., APIC[530:500] [Extended Interrupt [3:0] Local Vector Table]).

**Macro-op**: The front-end of the pipeline breaks instructions into macro-ops and transfers (dispatches) them to the back-end of the pipeline for scheduling and execution. See Software Optimization Guide.

**Master abort**: This is a PCI-defined term that is applied to transactions on other than PCI buses. It indicates that the transaction is terminated without affecting the intended target; Reads return all 1s; Writes are discarded; the master abort error code is returned in the response, if applicable; master abort error bits are set if applicable.

**Master or SMBus Master**: The device that initiates and terminates all communication and drives the clock, SCL.

**MAX**: MAX(integer expression list): Picks maximum integer or real value of comma separated list.

**MB**: Megabyte; 1024 KB.

**MCA**: Machine Check Architecture.

**MCAX**: Machine Check Architecture eXtensions.

**MergeEvent**: A PMC event that is capable of counter increments greater than 15, thus requiring merging a pair of even/odd performance monitors.

**Micro-op**: Processor schedulers break down macro-ops into sequences of even simpler instructions called micro-ops, each of which specifies a single primitive operation. See Software Optimization Guide.

**MIN**: MIN(integer expression list): Picks minimum integer or real value of comma separated list.

**MMIO**: Memory-Mapped Input-Output range. This is physical address space that is mapped to the IO functions such as the IO links or MMIO configuration.

**MMIO configuration**: Access to configuration space through memory space.

**MPB**: Microcode patch block.

**MSR:** The MSR, or x86 model specific register, physical register mnemonic format is of the form MSRXXXX\_XXXX, where XXXX\_XXXX is the hexadecimal MSR number. This space is accessed through x86 defined RDMSR and WRMSR instructions.

**MTRR:** Memory-type range register. The MTRRs specify the type of memory associated with various memory ranges.

**NBC:** NBC=(Cpuid Fn00000001\_EBX[LocalApicId[3:0]] == 0). Node Base Core. The lowest numbered core in the node.

**NTA:** Non-Temporal Access.

**OW:** Octword. An 128-bit value.

**P-state:** Performance state.

**PCICFG:** The PCICFG, or PCI defined configuration space, physical register mnemonic format is of the form DXFYxZZZ. Bus 0 is implied, X specifies the hexadecimal device number (this may be 1 or 2 digits). Y specifies the function number. ZZZ specifies the hexadecimal byte address (this may be 2 or 3 digits). Example: D18F2x40 specifies the register at bus 0, device 18h, function 2, and address 40h. If the mnemonic starts with B, then the physical mnemonic format is BWWDXYxZZZ where WW specifies the hexadecimal bus number (1 or 2 hex digits) or "XX" implying that the bus is relocatable. Example; BXXD00F6x40 specifies that the bus is relocatable, B0AD00F2x000 specifies that the bus is 0Ah.

**PCIe@:** PCI Express.

**PCS:** Physical Coding Sublayer.

**PEC:** Packet error code.

**physical mnemonic:** The register mnemonic that is formed based on the physical address used to access the register (e.g., D18F3x00). See 1.4.3.2 [Physical Mnemonic].

**PMC:** The PMC, or x86 performance monitor counter, physical register mnemonic format is any of the forms {PMcxXXX, L2IPMCxXXX, NBPMcxXXX}, where XXX is the performance monitor select.

**POR:** Power on reset.

**POW:** POW(base, exponent): POW(x,y) returns the value x to the power of y.

**PPIN:** Protected Processor Inventory Number.

**Processor:** System on Chip (SoC) covered by this PPR. See 1.8 [Processor Overview].

**PTE:** Page table entry.

**QW:** Quadword. A 64-bit value.

**RAS:** Reliability, availability and serviceability (industry term). See 3.1 [Machine Check Architecture].

**REFCLK:** Reference clock. Refers to the clock frequency (100 MHz) or the clock period (10 ns) depending on the context used.

**register instance parameter specifier:** A register instance parameter specifier is of the form \_register parameter name[register parameter value list] (e.g., The register instance parameter specifier \_dct[1:0] has a register parameter name of dct (The DCT PHY instance name) and a register parameter value list of "1:0" or 2 instances of DCT PHY).

**register instance specifier:** The register instance specifier exists when there is more than one instance for a register. The register instance specifier consists of one or more register instance parameter specifier (e.g., The register instance specifier \_dct[1:0]\_chiplet[BCST,3:0]\_pad[BCST,11:0] consists of 3 register instance parameter specifiers, \_dct[1:0], \_chiplet[BCST,3:0], and \_pad[BCST,11:0]).

**register name:** A name that connotes the function of the register.

**register namespace:** A namespace for which the register name must be unique. A register namespace indicates to which IP it belongs and an IP may have multiple namespaces. A namespace is a string that supports a list of "::-" separated names. The convention is for the list of names to be hierarchical, with the most significant name first and the least significant name last (e.g., Link::Phy::Rx is the RX component in the Link PHY).

**register parameter name:** A register parameter name is the name of the number of instances at some level of the logical hierarchy (e.g., The register parameter name dct specifies how many instances of the DCT PHY exist).

**register parameter value list:** The register parameter value list is the logical name for each instance of the register parameter name (e.g., For \_dct[1:0], there are 2 DCT PHY instances, with the logical names 0 and 1, but it should be noted that the logical names 0 and 1 can correspond to physical values other than 0 and 1). It is the purpose of the AddressMappingTable to map these register parameter values to physical address values for the register.

**Reserved-write-as-0:** Reads are undefined. Must always write 0.

**Reserved-write-as-1:** Reads are undefined. Must always write 1.

**ROUND:** ROUND(real expression): Rounds to the nearest integer; halfway rounds away from zero.

**RTS:** Remote temperature sensor, typical examples are ADM1032, LM99, MAX6657, EMC1002.

**SB-RMI:** Remote Management interface.

**SB-TSI:** Sideband Internal Temperature Sensor Interface. See APML.

**SBI:** Sideband interface.

**Shutdown:** A state in which the affected core waits for either INIT, RESET, or NMI. When shutdown state is entered, a shutdown special cycle is sent on the IO links.

**Slave or SMBus slave:** The slave cannot initiate SMBus communication and cannot drive the clock but can drive the data signal SDA and the alert signal ALERT\_L.

**SMAF:** System Management Action Field. This is the code passed from the SMC to the processors in STPLK assertion messages.

**SMI:** System management interrupt.

**SMM:** System Management Mode.

**SMT:** Simultaneous multithreading. See Core::X86::Cpuid::CoreId[ThreadsPerCore].

**Speculative event:** A performance monitor event counter that counts all occurrences of the event even if the event occurs during speculative code execution.

**SSC:** Spread Spectrum Clocking.

**SVM:** Secure virtual machine.

**TCC:** Temperature calculation circuit.

**Tctl:** Processor temperature control value.

**TDC:** Thermal Design Current.

**TDP:** Thermal Design Power. A power consumption parameter that is used in conjunction with thermal specifications to design appropriate cooling solutions for the processor.

**Thread:** One architectural context for instruction execution.

**TOM2:** Top of extended Memory.

**TSI:** Temperature sensor interface.

**TSM:** Temperature sensor macro.

**UMI:** Unified Media Interface. The link between the processor and the FCH.

**UNIT:** UNIT(register field reference): Input operand is a register field reference that contains a valid values table that defines a value with a unit (e.g., clocks, ns, ms, etc.). This function takes the value in the register field and returns the value associated with the unit (e.g., If the field had a valid value definition where 1010b was defined as 5 ns). Then if the field had the value of 1010b, then UNIT() would return the value 5.

**Unpredictable:** The behavior of both reads and writes is unpredictable.

**VID:** Voltage level identifier.

**VMPL:** Virtual Machine Privilege Level.

**Volatile:** Indicates that a register field value may be modified by hardware, firmware, or microcode when fetching the first instruction and/or might have read or write side effects. No read may depend on the results of a previous read and no write may be omitted based on the value of a previous read or write. Not volatile indicates that software may service a read from the results of a previous read and that a write may be dropped if it's value matches the value previously read or written.

**Warm reset:** RESET\_L is asserted only (while PWROK stays high).

**WDT:** Watchdog timer. A timer that detects activity and triggers an error if a specified period of time expires without the activity.

**WRIG:** Writes Ignored.

**Write-0-only:** Writing a 0 clears to a 0; Writing a 1 has no effect. If not associated with Read, then reads are undefined.

**Write-1-only:** Writing a 1 sets to a 1; Writing a 0 has no effect. If not associated with Read, then reads are undefined.

**Write-1-to-clear:** Writing a 1 clears to a 0; Writing a 0 has no effect. If not associated with Read, then reads are undefined.

**Write-once:** Capable of being written once; all subsequent writes have no effect. If not associated with Read, then reads are undefined.

**X2APICEN:** x2 APIC is enabled. X2APICEN =

(Core::X86::Msrr::APIC\_BAR[ApicEn] && Core::X86::Msrr::APIC\_BAR[x2ApicEn]).

**XBAR:** Cross bar; command packet switch.