Processor Programming Reference (PPR) for AMD Family 17h Models 01h,08h, Revision B2 Processors

Legal Notices

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 this document, 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 this document, 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 this document. 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.
3DNow! is a trademark of Advanced Micro Devices, Incorporated.
AGESA is a trademark of Advanced Micro Devices, Incorporated.
AMD Secure Encrypted Virtualization is a trademark of Advanced Micro Devices, Incorporated.
AMD Secure Memory Encryption is a trademark of Advanced Micro Devices, Incorporated.
AMD Virtualization is a trademark of Advanced Micro Devices, Incorporated.
ARM is a registered trademark of ARM Limited.
DesignWare is a registered trademark of Synopsys Incorporated.
MMX is a trademark of Intel Corporation.
Microsoft is a registered trademark of Microsoft Corporation.
PCI Express is a registered trademark of PCI-SIG Corporation.
PCIe is a registered trademark of PCI-SIG Corporation.
Ultrabook is a registered trademark of Intel 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

Table of Contents

List of Figures

List of Tables

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
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.
docACPI	Advanced Configuration and Power Interface (ACPI) Specification. http://www.acpi.info.
docIOMMU	AMD I/O Virtualization Technology Specification, order# 48882.
docI2C	I2C Bus Specification. http://www.nxp.com/documents/user_manual/UM10204.pdf
docJEDEC	JEDEC Standards. http://www.jedec.org.
docPCIe	PCI Express® Specification. http://www.pcisig.org.
docPCIlb	PCI Local Bus Specification. http://www.pcisig.org.
docRAS	RAS Feature Enablement for AMD Family 17h Models 00h-0Fh, order# 55987.
docRevG	Revision Guide for AMD Family 17h Models 00h-0Fh Processors, order# 55449.
docAM4	Socket AM4 Processor Functional Data Sheet, order# 55509.

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 Conventions

1.3.1 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).

1.3.2 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**
ABS	ABS(integer expression): Remove sign from signed value.
FLOOR	FLOOR(integer expression): Rounds real number down to nearest integer.
CEIL	CEIL(real expression): Rounds real number up to nearest integer.
MIN	MIN(integer expression list): Picks minimum integer or real value of comma separated list.
MAX	MAX(integer expression list): Picks maximum integer or real value of comma separated list.
COUNT	COUNT(integer expression): Returns the number of binary 1's in the integer.
ROUND	ROUND(real expression): Rounds to the nearest integer; halfway rounds away from zero.
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.
POW	POW(base, exponent): POW(x,y) returns the value x to the power of y.

1.3.2.1 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

1.3.3 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**
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 1.3.3.1 [Logical Mnemonic].
physical mnemonic	The register mnemonic that is formed based on the physical address used to access the register (e.g., D18F3x00). See 1.3.3.2 [Physical Mnemonic].

1.3.3.1 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
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 name	A name that cannotes the function of the register.
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 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 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.

1.3.3.2 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
PCICFG	The PCICFG, or PCI defined configuration space, physical register mnemonic format is of the form DXFYxZZZ.
BAR	The BAR, or base address register, physical register mnemonic format is of the form PREFIXxZZZ.
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.
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.
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.

1.3.4 Register Format

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

1.3.4.1 A Register is a group of Register Instances

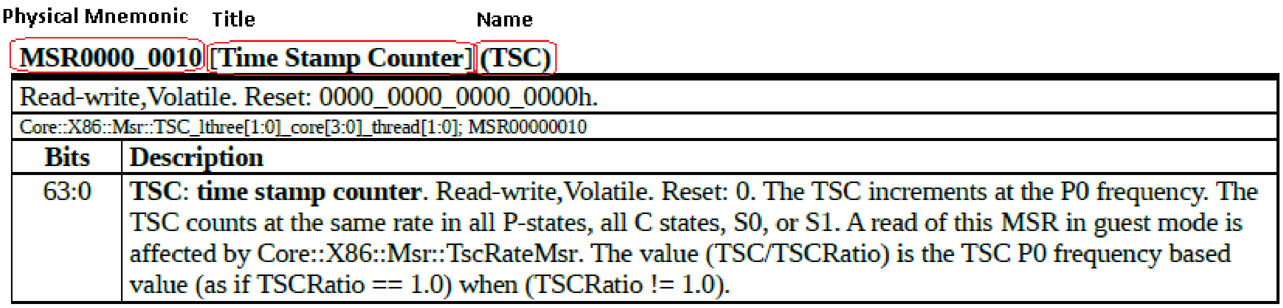
All instances of a register:

- 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 1.3.4.10 [Field Access Type].
  - Reset. See 1.3.4.11 [Field Reset].
  - Init. See 1.3.4.12 [Field Initialization].
  - Check. See 1.3.4.13 [Field Check].

1.3.4.2 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 1.3.4.5 [Register Instance Table].
2. The register title in brackets.
3. The register name in parenthesis.



MSR0000_0010 [Time Stamp Counter] (TSC)

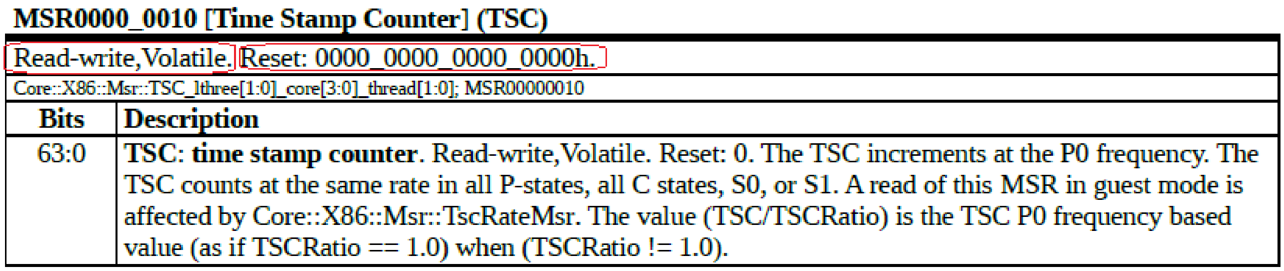
Read-write,Volatile. Reset: 0000_0000_0000_0000h.	
Core::X86::Msr::TSC_lthree[1:0]_core[3:0]_thread[1:0]; MSR00000010	
Bits	**Description**
63:0	**TSC**: **time stamp counter**. Read-write,Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all 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 1: Register Physical Mnemonic, Title, and Name

1.3.4.3 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 1.3.4.10 [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 1.3.4.11 [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.



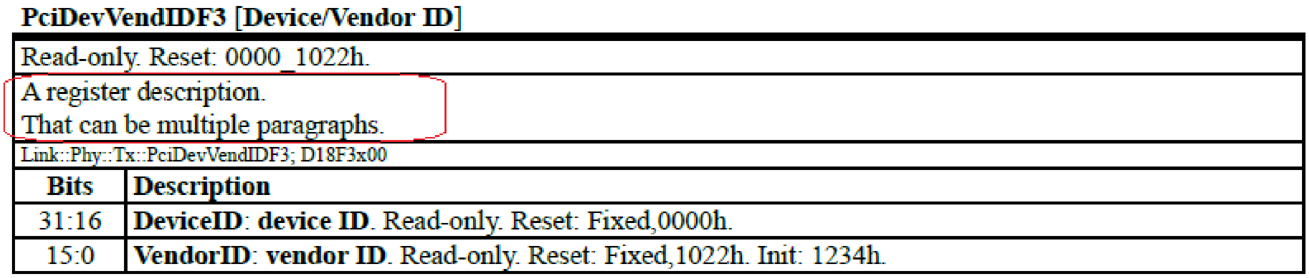
MSR0000_0010 [Time Stamp Counter] (TSC)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.	
Core::X86::Msr::TSC_lthree[1:0]_core[3:0]_thread[1:0]; MSR00000010	
Bits	**Description**
63:0	**TSC**: **time stamp counter**. 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 2: Full Width Register Attributes

1.3.4.4 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 3: Register Description

1.3.4.5 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:

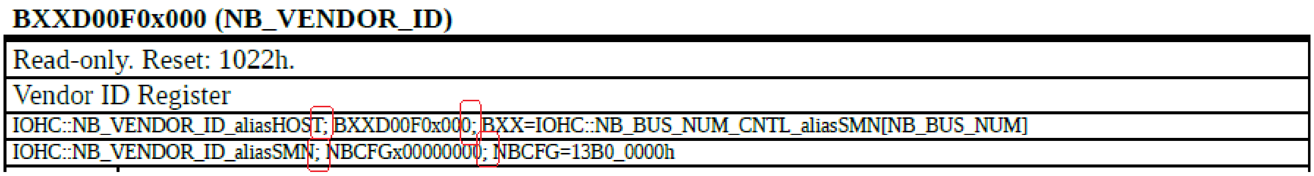
- Each row describes the access method of one or more register instances.
- If a row describes two or more instances, then the logical instance range, left to right, corresponds to the physical range, left to right.
- The absence of register instance rows indicates that the register exists for documentation 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.

1.3.4.5.1 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 1.3.3.1 [Logical Mnemonic].
- The text after the first semicolon is the physical mnemonic.
  - See 1.3.3.2 [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 4: Register Instance Table: Content Ordering in a Row

1.3.4.5.2 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 FFF1x000000088_x000B0001.

1.3.4.5.3 MSR Access Method

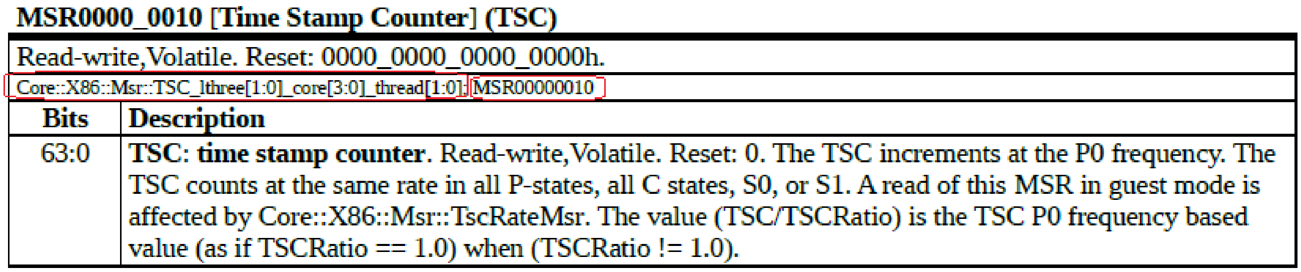
The MSR parameters {lthree,core,thread} are implied by the identity of the core on which the RDMSR/WRMSR is being executed, and therefore are not represented in the physical mnemonic.

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.

1.3.4.5.3.1 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

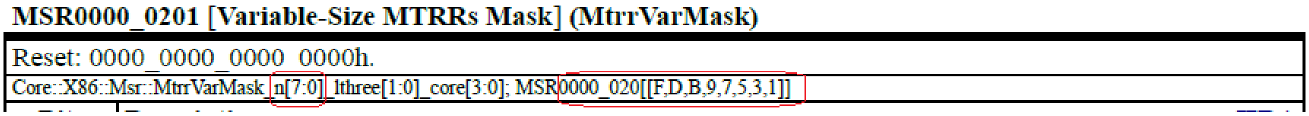
Bits	Description
63:0	**TSC: time stamp counter**. Read-write,Volatile. Reset: 0. The TSC increments at the P0 frequency. The TSC counts at the same rate in all P-states, all C states, S0, or S1. A read of this MSR in guest mode is affected by Core::X86::Msr::TscRateMsr. The value (TSC/TSCRatio) is the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

Figure 5: Register Instance Table: MSR Example

1.3.4.5.3.2 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_020[[F,D,B,9,7,5,3,1]]

Figure 6: Register Instance Table: MSR Range Example

1.3.4.5.4 BAR Access Method

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

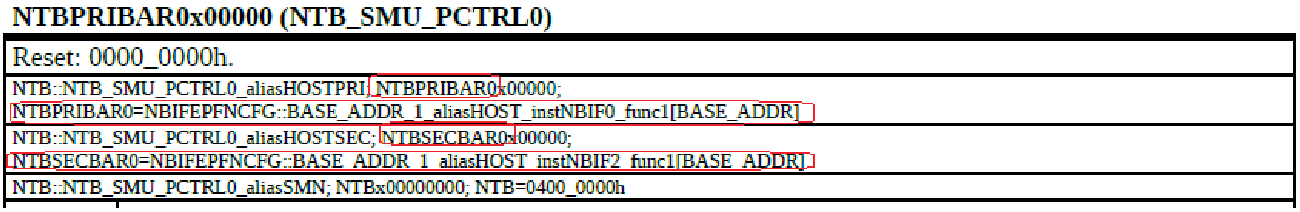
- Example: APICx0000. The BAR prefix is "APIC".

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

1.3.4.5.4.1 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].



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 7: Register Instance Table: BAR as Register Reference

1.3.4.5.5 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.

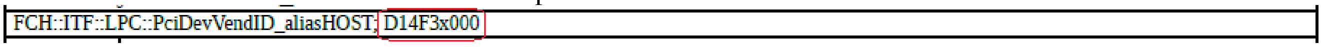
Example:

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

1.3.4.5.5.1 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

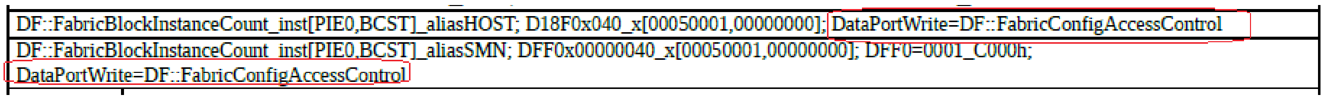
Figure 8: Register Instance Table: Bus Implied to be 00h

1.3.4.5.6 Data Port Access Method

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

Example:

- The data port select value follows the "_x".
- The data port select register follows the "DataPortWrite=".



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 9: Register Instance Table: Data Port Select

1.3.4.6 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.

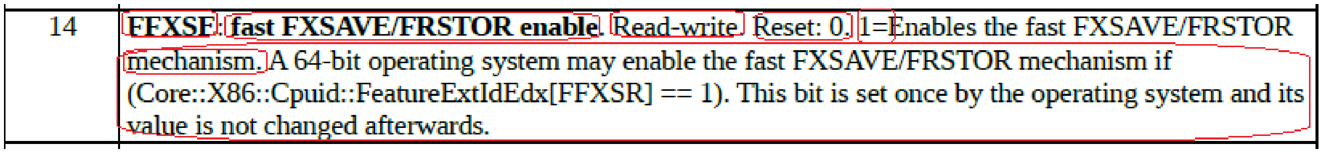
There are 2 field definition formats, simple and complex. If the description can be described in the simple one paragraph format then the simple format is used, else the complex format is used.

1.3.4.7 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 1.3.4.9 [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 1.3.4.10 [Field Access Type].
   - In the example figure the access type is "Read-write".
4. Field Reset. See 1.3.4.11 [Field Reset].
   - In the example figure the reset is warm reset and "0".
5. Field Init. See 1.3.4.12 [Field Initialization].
6. Field Check. See 1.3.4.13 [Field Check].
7. Field Valid Values, if the valid values are single bit (E.g. 0=, 1=). See 1.3.4.14 [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 don't exist are omitted.



14	**FFXSE**: **fast FXSAVE/FRSTOR enable**. 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 10: Simple Register Field Example

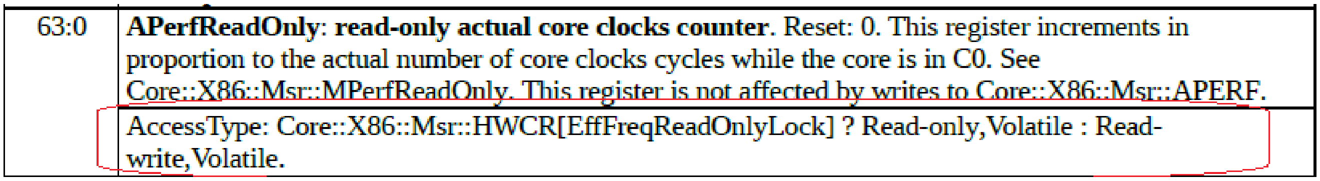
1.3.4.8 Complex Register Field Format

Content that can't 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)

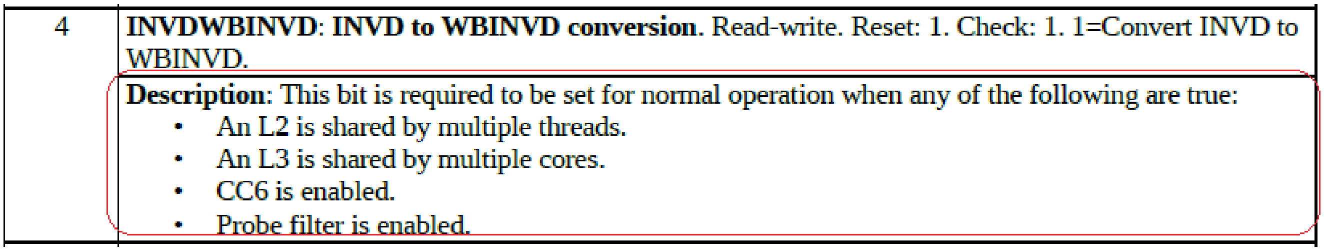
The following figure highlights a complex access type specification.



63:0	**APerfReadOnly**: **read-only actual core clocks counter**. 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 11: Register Field Sub-Row for {Reset,AccessType,Init,Check}

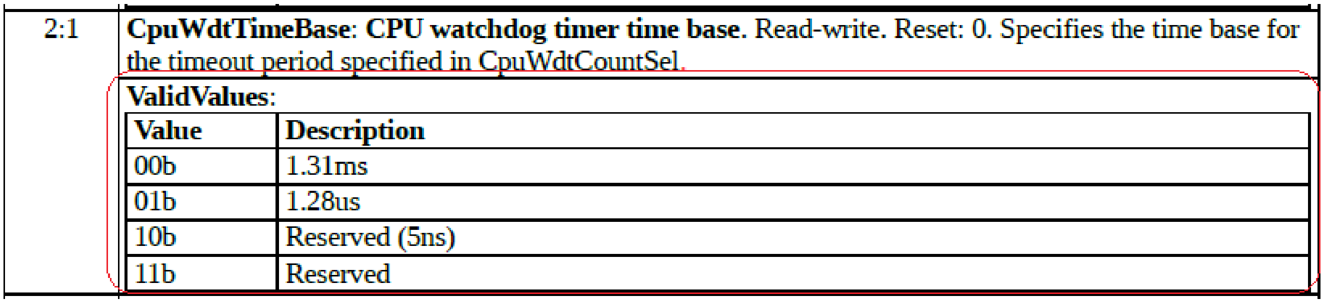
The following figure highlights a complex description specification.



4	**INVDWBINVD**: **INVD to WBINVD conversion**. Read-write. Reset: 1. Check: 1. 1=Convert INVD to WBINVD.
	Description: This bit is required to be set for normal operation when any of the following are true: • An L2 is shared by multiple threads. • An L3 is shared by multiple cores. • CC6 is enabled. • Probe filter is enabled.

Figure 12: 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.



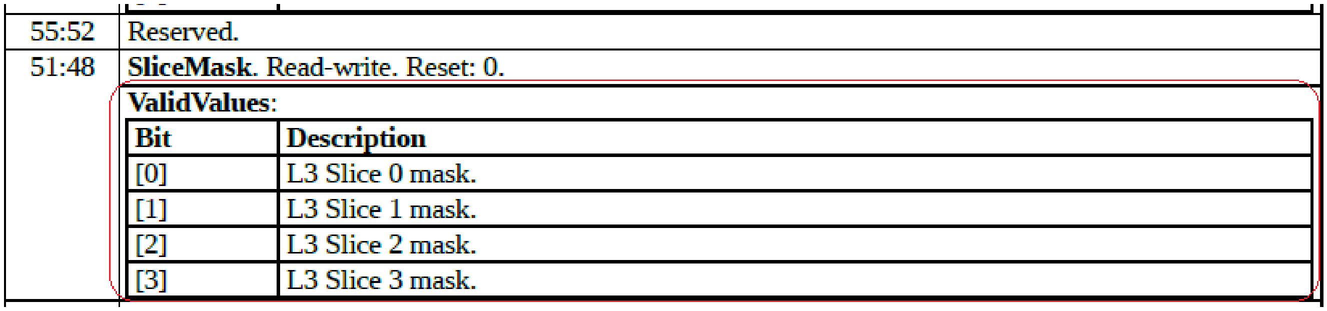
2:1 **CpuWdtTimeBase**: **CPU watchdog timer time base**. Read-write. Reset: 0. Specifies the time base for the timeout period specified in CpuWdtCountSel.

ValidValues:

Value	Description
00b	1.31ms
01b	1.28us
10b	Reserved (5ns)
11b	Reserved

Figure 13: 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	**SliceMask**. Read-write. Reset: 0. **ValidValues**:

Bit	Description
[0]	L3 Slice 0 mask.
[1]	L3 Slice 1 mask.
[2]	L3 Slice 2 mask.
[3]	L3 Slice 3 mask.

Figure 14: Register Field Sub-Row for Valid Bit Table

1.3.4.9 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 "Reserved-write-as-read".

- The Reserved-write-as-read access type is:
  - Reads must not depend on the read value.
  - Writes must only write the value that was read.

1.3.4.10 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
Read-only	Readable; writes are ignored.
Read-write	Readable and writable.
Read	Readable; must be associated with one of the following {Write-once, Write-1-only, Write-1-to-clear, Error-on-write}.
Write-once	Capable of being written once; all subsequent writes have no effect. If not associated with Read, then reads are undefined.
Write-only	Writable. 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-0-only	Writing a 0 clears to a 0; Writing a 1 has no effect. If not associated with Read, then reads are undefined.
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.
Inaccessible	Not readable or writable (e.g., Hide ? Inaccessible : Read-Write).
Configurable	Indicates that the access type is configurable as described by the documentation.
Unpredictable	The behavior of both reads and writes is unpredictable.
Reserved-write-as-1	Reads are undefined. Must always write 1.

Reserved-write-as-0	Reads are undefined. Must always write 0.
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.

1.3.4.10.1 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

1.3.4.11 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 value applies at all time.

1.3.4.12 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.

1.3.4.13 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.

1.3.4.14 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 Definitions

Table 11: Definitions

Term	Description
AGESA™	AMD Generic Encapsulated Software Architecture.
APML	Advanced Platform Management Link.
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).
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.
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.
COF	Current operating frequency of a given clock domain.
Cold reset	PWROK is deasserted and RESET_L is asserted.
DID	Divisor Identifier. Specifies the post-PLL divisor used to reduce the COF.
Doubleword	A 32-bit value.
DW	Doubleword.
ECS	Extended Configuration Space.
EDC	Electrical design current. Indicates the maximum current the voltage rail can demand for a short, thermally insignificant time.
FCH	The integrated platform subsystem that contains the IO interfaces and bridges them to the system BIOS. Previously included in the Southbridge.
FDS	Functional Data Sheet. There is one FDS for each package type. See docSAM4.
FID	Frequency Identifier. Specifies the PLL frequency multiplier for a given clock domain.
GB	Gbyte or Gigabyte; 1,073,741,824 bytes.
GT/s	Giga-Transfers per second.
HTC	Hardware Thermal Control.
HTC-active state	Hardware-controlled lower-power, lower performance state used to reduce temperature.
IO configuration	Access to configuration space though IO ports CF8h and CFCh.
IP	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.
KB	Kbyte or Kilobyte; 1024 bytes.
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; write are discarded; the master abort error code is returned in the response, if applicable; master abort error bits are set if applicable.
MB	Megabyte; 1024 KB.
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.

Node	A node, is an integrated circuit device that includes one to 8 cores (one or two Core Complexes).
OW	Octword. An 128-bit value.
Processor	A package containing one or more Nodes. See Node.
QW	Quadword. A 64-bit value.
RX	Receiver.
REFCLK	Reference clock. Refers to the clock frequency (100 MHz) or the clock period (10 ns) depending on the context used.
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.
SMAF	System Management Action Field. This is the code passed from the SMC to the processors in STPCLK assertion messages.
SMC	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.
Speculative event	A performance monitor event counter that counts all occurrences of the event even if the event occurs during speculative code execution.
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.
Token	A scheduler entry used in various DF queues to track outstanding requests.
TX	Transmitter.
UMI	Unified Media Interface. The link between the processor and the FCH.
VDD	Main power supply to the processor core logic.
VID	Voltage level identifier.
Warm reset	RESET_L is asserted only (while PWROK stays high).
XBAR	Cross bar; command packet switch.
PCIe®	PCI Express.

1.5 Changes Between Revisions and Product Variations

1.5.1 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.

1.6 Package

1.6.1 Package type

The following packages are supported.

Table 12: Package Definitions

Term	Description

AM4	Desktop, single die, single socket. For client platform DDR4. AM4 = (Core::X86::Cpuid::BrandId[PkgType] == 02h).
SP3	Server, four die MCM, single and dual socket.
SP3r2	High performance desktop, 4-die, single socket, lidded LGA.

1.7 Processor Overview

1.7.1 Features

Family 17h Models 00h-0Fh are a microprocessor System-On-a-Chip (SOC) featuring AMD x86 cores. It also uses a scalable data fabric (SDF), integrated IO, and integrated southbridge control hub, SCH where no supporting chipset is necessary.

- CPU:
  - 2 Core Complexes (CCX). The Core represents the x86 ISA core from AMD designed for FX and 7th generation APU offerings.
    - Supports Simultaneous Multithreading over previous generations clusters.
  - Each core complex consists of:
    - 4 cores where each core may run in single-thread mode (1T) or two-thread SMT mode (2T) for a total of up to 8 threads per complex
    - 512KB of L2 per core for a total of 2MB L2 per complex
      - 4MB L2 total
    - 8MB of L3 shared across all cores within the complex
      - 16MB L3 total
- Scalable Data Fabric. This provides the data path that connects the compute complexes, the I/O interfaces, and the memory interfaces to each other.
  - Handles request, response, and data traffic
  - Handles probe traffic to facilitate coherency, including a probe filter supporting up to 512GB per DRAM channel
  - Handles interrupt request routing (APIC)
- Memory interface
  - 2 Unified Memory Controllers (UMC), each supporting one DRAM channel
  - 2 DDR4 PHYs. Each PHY supports:
    - 64-bit data plus ECC
    - 1 DRAM channel per PHY
    - 2 DIMMs per channel
      - DDR4 transfer rates from 1333MT/s to 3200MT/s
      - UDIMM,SODIMM
- PSP and SMU
  - MP0 (PSP) and MP1 (SMU) microcontrollers
    - This document refers to the AMD Secure Processor technology as Platform Security Processor (PSP).
  - Thermal monitoring
  - Fuses
  - Clock control
- NBIO
  - 2 SYSHUBs
  - 1 IOHUB
  - Two 8x16 PCIe® controllers supporting Gen1/Gen2/Gen3. Note that SATA Express is supported by combining an x2 PCIe® port and two SATA ports on the same 2 lanes.
- Enterprise 12G (E12G) Combo PHYs, PCS, and UPI muxing

    - 6 x4 PHYs plus 5 x2 PHYs
    - PHYs can support the following controller types: PCIe®, WAFL, xGMI, SATA, and Ethernet (SGMII 1000/100/10, 10GBASE-KR, 1000BASE-KX protocols). In addition, SATA Express can be supported by combining PCIe® and SATA controllers on the same lanes with a GPIO for a device to indicate its controller type.
    - PHY muxing is provided that allows different package or board configurations to enable a single PHY to support functionality from multiple on-die controllers
- Fusion Controller Hub (FCH) or southbridge (SB))
    - ACPI
    - CLKGEN/CGPLL for refclk generation
    - eMMC
    - GPIOs (varying number depending on muxing)
        - (6 ports)
    - LPC
    - Real-Time Clock (RTC)
    - SMBus (2 ports)
    - SPI/eSPI
    - UART (4 ports)
- Azalia
    - High Definition Audio
- Ethernet complex
    - Up to 4 lanes of 10/100/1000 SGMII, or 10GBASE-KR, or 1000BASE-KX Ethernet operation
    - 2 instances of a "lite" controller configuration
    - 2 instances of a "heavy" controller configuration
- SATA
    - Up to 8 lanes of SATA Gen1/Gen2/Gen3, also provides the legacy SATA support for SATAe ports
- SGPIO
- USB3.0
    - 4 ports of USB3 SuperSpeed
    - includes support for legacy USB speeds

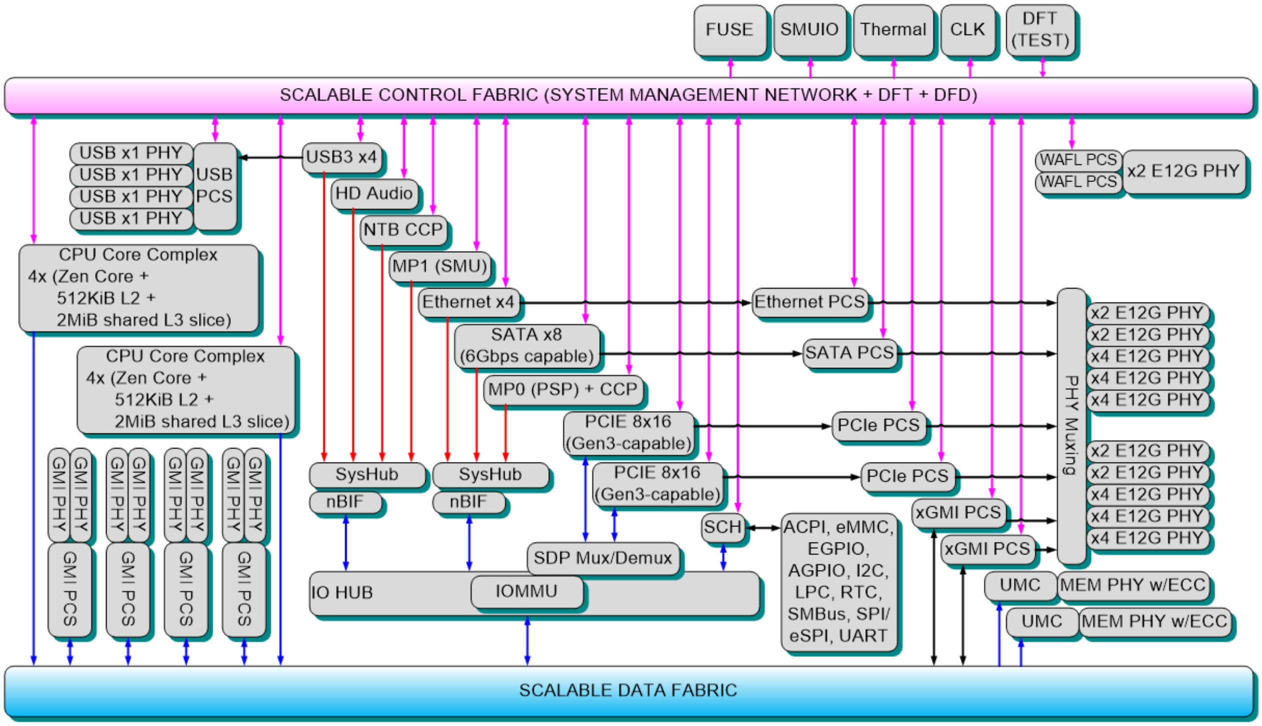


Figure 15: Family 17h Models 00h-0Fh Processor Overview

1.8 System Overview

1.8.1 AM4 Desktop

AM4 is a single-socket client infrastructure supporting DDR4 and PCIe® for non-coherent I/O communication. The AM4 package is a lidded µPGA package that supports AMD Family 17h Models 00h-0Fh die, and Family 17h Models 70h - 7Fh,.

Table 13: AM4 1P Capabilities

	AM4 **1P Configuration**
Module Type	Single or multi-die, micro pin grid array common socket infrastructure with other AM4 products
Cores / module	8
Memory channel/module	2
Max DIMMs/channel	2
DIMM Type	1.2V up to DDR4-3200
Combo links/module (note1)	PHY groupings of 16 lanes may each have a maximum of 8 PCIe® ports, where a port consists of a power-of-2 lanes (x1, x2, x4, x8, x16) or a SATA Express port
Max PCIe®/module	24 lanes: 16 for dGPU, 4 for IO expander, 4 for storage (NVMe or 2 ports SATA Express)
Max SATA/ module (note2)	Up to 4 Gen 3
Native I/O	USB3/2, SPI, LPC, I2C, RTC, Power control, etc.
Notes: 1: Combo links can take the form of PCIe®, SATA, SATA Express with configuration restrictions. 2: These functions are in lieu of PCIe® on those ports (e.g., a group of 8 SATA displaces 8 PCIe® lanes).	

1.8.2 SP3 MCM Server Single-Socket

SP3-based one socket systems target a new paradigm of high end single socket commercial servers. These servers have a large number of memory channels, memory capacity, and I/O ports and target the following form factors:

- 1U traditional rack servers.
- Blade servers.
- Multi node, 1P "twins" type of shared infrastructure servers.
- EATX or EATX+ servers.

The characteristics and capabilities of the SP3 product in 1 socket systems are shown in the following table:

Table 14: SP3 1P Capabilities

	SP3 **1P Configuration**
Module Type	4 die MCM socketed LGA
Module size, pad pitch	58.5 x 75.4 @ 1.00 x 0.87mm pitch

Socket size	78.9 x 119.3 mm, heatsink actuated
Cores / module	32 (8 per die)
Memory chan/module	2 (2 per die)
Max DIMMs/channel	2
DIMM Type	R/LR/NV DIMM, 3DS
Combo links/module (note1)	Eight – 16 bit links (2 per die)
xGMI links (16 bit)	N/A
Max PCIe®/module	128 lanes (32 per die)
Max SATA/module (note2)	32
Max 10GigE/module (note2)	16 (note3)
Native I/O	USB3/2, SPI, LPC, UART, I2C, RTC, Power control, etc.
Notes: 1: Combo links can take the form of PCIe®, SATA, SATA Express, 10GBASE-KR, 1000BASE-KX, SGMII with configuration restrictions. 2: These functions are in lieu of PCIe® on those ports (e.g., a group of 8 SATA displaces 8 PCIe® lanes). 3: Half of the Ethernet ports have 16 TX/RX DMA queues (WHQL compliant for 10GBASE-KR or SGMII or 1000BASE-KX). Half have 2 TX/RX DMA queues (WHQL compliant for SGMII or 1000BASE-KX only).	

1.8.2.1 SP3 1P Coherent Interconnect Topology

Referring to 1P System Block Diagram below: Each of the 4 dies on the SP3 MCM is interconnected by one GMI link on the substrate, creating a fully connected 4-die topology. Since this is deployed as a 1P server, xGMI mode is not utilized. Instead, the links can be deployed to double the I/O connectivity.

1.8.2.2 SP3 1P Memory Support

Each die on the MCM system generates two memory channels resulting in 16 memory channels in a two socket system. Each channel can accommodate two DDR4 DIMM connectors. DDR3 is not supported. The supported DIMM types are:

- One and two rank DDR4 RDIMMs.
- Four and eight physical rank DDR4 LRDIMMs.
- DDR4 3DS DIMMs.
- NVDIMM-Ns.

1.8.2.3 SP3 1P I/O Support

The I/O interfaces for the Family 17h Models 00h-0Fh die are configurable and extremely flexible. Each die generates two 16 bit "combo" links, one is type A and the other is type B.

Type A links can be configured to support:

- One 16-lane PCIe® Gen3 controller with 8 ports.
- Up to eight SATA 3 ports.
- Up to four 10Gbps-capable Ethernet ports.
- Up to two SATA Express ports.

Type B links can support:

- One 16 lane PCIe® gen 3 controller with 8 ports.

This configuration capability of the links is shown in the Link Capabilities figure below.

In one socket systems, there are 8 links, 4 type A and 4 type B, all of which can be used as I/O. As an example, a one socket system that maximizes PCIe® connectivity can support up to 128 lanes of PCIe® Gen3. This can be divided up into smaller link widths as long as there are no more than 8 ports per 16-lane group.

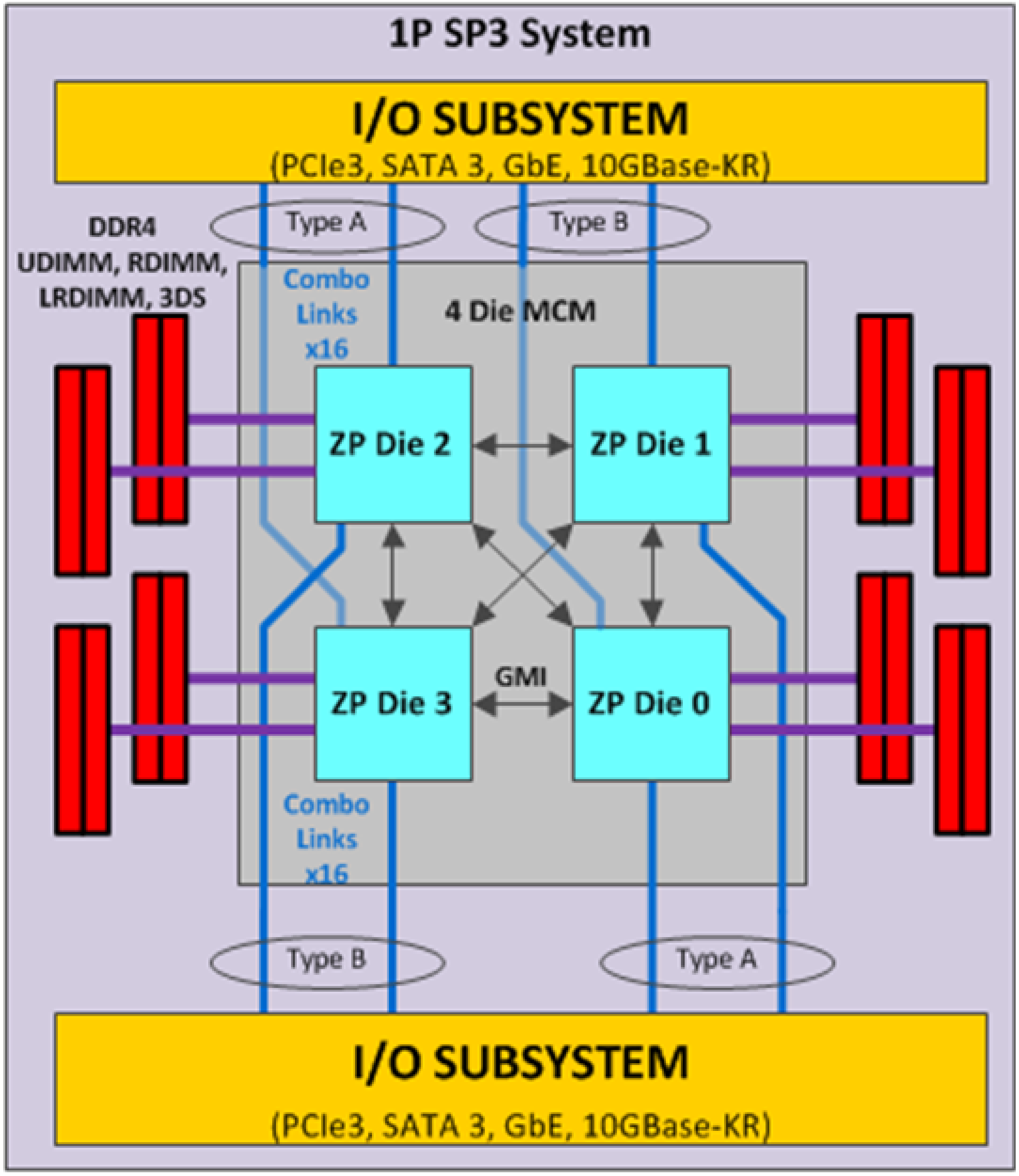


Figure 16: 1P System Block Diagram

1.8.3 Mixed Processor Revision Support

AMD Family 17h processors with different OPNs or different revisions cannot be mixed in a multiprocessor system. If an unsupported configuration is detected, BIOS should configure the BSP as a single processor system and signal an error.

2 Core Complex (CCX)

2.1 Processor x86 Core

2.1.1 Core Functional Information

2.1.2 Core Definitions

Table 15: Definitions

Term	Description
CCX	Core Complex where more than one core shares L3 resources.
Core	The instruction execution unit of the processor when the term Core is used in a x86 core context.
CoreCOF	Core current operating frequency in MHz. CoreCOF = (Core::X86::Msr::PStateDef[CpuFid[7:0]]/Core::X86::Msr::PStateDef[CpuDfsId])*200.
CPL	Current Privilege Level of the running task when the term CPL is used in a x86 core context.
CpuCoreNum	Specifies the core number.
IBS	Instruction based sampling.
IO configuration	Access to configuration space through IO ports CF8h and CFCh.
IORR	IO range register.
L1 cache	The level 1 caches (instruction cache and the data cache).
L2 cache	The level 2 caches.
L3	Level 3 Cache. The L3 term is also in Addrmaps to enumerate CCX units.
L3 cache	Level 3 Cache.
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.
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]).
Micro-op	Micro-op. Instructions have variable-length encoding and many perform multiple primitive operations. The processor does not execute these complex instructions directly, but, instead, decodes them internally into simpler fixed-length instructions called macro-ops. Processor schedulers subsequently break down macro-ops into sequences of even simpler instructions called micro-ops, each of which specifies a single primitive operation.
MTRR	Memory-type range register. The MTRRs specify the type of memory associated with various memory ranges.
NTA	Non-Temporal Access.
PTE	Page table entry.
SMI	System management interrupt.
Speculative event	A performance monitor event counter that counts all occurrences of the event even if the event occurs during speculative code execution.
SVM	Secure virtual machine.
BSC	Boot strap core. Core 0 of the BSP.
BSP	Boot strap processor.

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.
CMP	Specifies the core number.
#GP	A general-protection exception.
#GP(0)	Notation indicating a general-protection exception (#GP) with error code of 0.
NBC	NBC = (CPUID Fn00000001_EBX[LocalApicId[3:0]]==0). Node Base Core. The lowest numbered core in the node.
SMM	System Management Mode.
SMT	Simultaneous multithreading. See Core::X86::Cpuid::CoreId[ThreadsPerCore].
Thread	One architectural context for instruction execution.
WDT	Watchdog timer. A timer that detects activity and triggers an error if a specified period of time expires without the activity.

2.1.3 Secure Virtual Machine Mode (SVM)

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

2.1.3.1 BIOS support for SVM Disable

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

2.1.3.1.1 Enable AMD Virtualization™

- Core::X86::Msr::VM_CR[SvmeDisable] = 0.
- Core::X86::Msr::VM_CR[Lock] = 1.
- Core::X86::Msr::SvmLockKey[SvmLockKey] = 0000_0000_0000_0000h.

2.1.3.1.2 Disable AMD Virtualization™

- Core::X86::Msr::SvmLockKey[SvmLockKey] = 0000_0000_0000_0000h.
- Core::X86::Msr::VM_CR[SvmeDisable] = 1.
- Core::X86::Msr::VM_CR[Lock] = 1.

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

2.1.3.1.3 Disable AMD Virtualization™, with a user supplied key

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

2.1.4 Memory Encryption

For details of the memory encryption, see docAPM2 section Secure Encrypted Virtualization. See docAPM2 section Enabling Memory Encryption Extensions for details about enabling memory encryption extensions.

2.1.5 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::Msr::MPERF and the write of Core::X86::Msr::APERF in step 1 or between the read of Core::X86::Msr::MPERF and the read of Core::X86::Msr::APERF in step 2.
- The behavior of Core::X86::Msr::MPERF and Core::X86::Msr::APERF may be modified by Core::X86::Msr::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::Msr::MPERF and Core::X86::Msr::APERF software executes only MOV instructions, and no more than 3 MOV instructions, between the two RDMSR or WRMSR instructions.
    - Core::X86::Msr::MPERF and Core::X86::Msr::APERF are invalid if an overflow occurs.

2.1.6 Address Space

2.1.6.1 Virtual Address Space

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

2.1.6.2 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].

2.1.6.3 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.

2.1.6.3.1 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 NB: 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.

2.1.6.3.1.1 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 APM2 chapter titled "Memory System", sections "Memory-Type Range Registers" and "Page-Attribute Table Mechanism".
    - See the APM2 chapter titled "Nested Paging", section "Combining Memory Types, MTRRs".
    - See Core::X86::Msr::MTRRdefType, Core::X86::Msr::MtrrVarBase, Core::X86::Msr::MtrrVarMask, Core::X86::Msr::MtrrFix_64K and Core::X86::Msr::MtrrFix_16K_0 through Core::X86::Msr::MtrrFix_4K_7.
2. TSeg & ASeg SMM mechanism. (see Core::X86::Msr::SMMAddr and Core::X86::Msr::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::Msr::SMMMask[AValid] && "The address falls within the ASeg region") || (Core::X86::Msr::SMMMask[TValid] && "The address falls within the TSeg region"))) then MemType=CD.

2.1.7 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 2.1.7.3 [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::Msr::HWCR[IoCfgGpFault].
    - SMI trapping for these accesses is specified by Core::X86::Msr::SMI_ON_IO_TRAP_CTL_STS and Core::X86::Msr::SMI_ON_IO_TRAP.
- MMIO configuration: configuration space is a region of memory space.
    - The base address and size of this range is specified by Core::X86::Msr::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, 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.

2.1.7.1 MMIO Configuration Coding Requirements

MMIO configuration space accesses must use the uncacheable (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

2.1.7.2 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.

2.1.7.3 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.

2.1.8 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 2.1.7 [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	**ConfigEn**: **configuration space enable**. 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	**ExtRegNo**: **extended register number**. 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	**BusNo**: **bus number**. Read-write. Reset: 00h. Specifies the bus number of the configuration cycle.
15:11	**Device**: **device number**. Read-write. Reset: 00h. Specifies the device number of the configuration cycle.
10:8	**Function**. Read-write. Reset: 0h. Specifies the function number of the configuration cycle.
7:2	**RegNo**: **register address**. 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	**Data**. Read-write. Reset: 0000_0000h. See IO::IoCfgAddr.

2.1.9 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.

2.1.10 Register Sharing

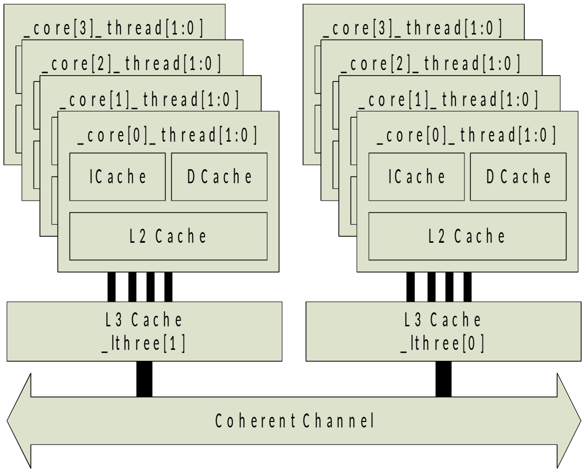
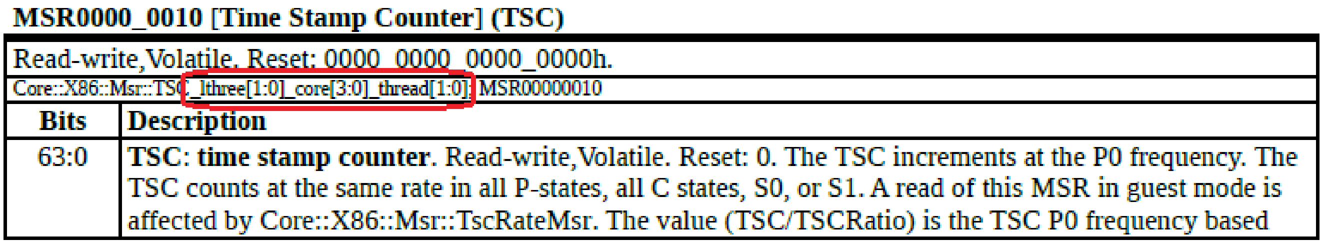


Figure 17: Register Sharing Domains



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	**TSC: time stamp counter**. 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

Figure 18: Instance Parameters

Instances of core registers are designated as lthree[n:0]_core[n:0]_thread[1:0]. Core registers may be shared at various levels of hierarchy as one register instance per die, 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. Software must coordinate writing to shared registers with other threads in the same sharing hierarchy level.

2.1.11 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 2xCLKIN; the APIC timer may increment in units of between 1 and 8.

2.1.12 Interrupts

2.1.12.1 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.

2.1.12.1.1 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.

2.1.12.1.2 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::Msr::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::Msr::SMM_BASE[SmmBase]. Important offsets to the base address pointer are:

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

2.1.12.1.3 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 is stored in Core::X86::Smm::LocalSmiStatus.

2.1.12.1.4 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 16: 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, and 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.

2.1.12.1.5 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 17: SMM Save State

Offset	Size	Contents		Access
FE00h	Word	ES	Selector	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	Doubleword		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	Doubleword		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	Doubleword		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	Doubleword		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	Doubleword	Core::X86::Smm::TrapOffset [SMM IO Trap Offset]		Read-only
FEC4	Doubleword	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	Read-write
FF90h	Quadword	R13	
FF98h	Quadword	R12	
FFA0h	Quadword	R11	
FFA8h	Quadword	R10	
FFB0h	Quadword	R9	
FFB8h	Quadword	R8	
FFC0h	Quadword	RDI	
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).

2.1.12.1.6 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	**Port**: **trapped IO port address**. Read-only,Volatile. Reset: 0000h. This provides the address of the IO instruction.
15:12	**BPR**: **IO breakpoint match**. Read-only,Volatile. Reset: 0h.
11	**TF**: **EFLAGS TF value**. Read-only,Volatile. Reset: 0.
10:7	Reserved.
6	**SZ32**: **size 32 bits**. Read-only,Volatile. Reset: 0. 1=Port access was 32 bits.
5	**SZ16**: **size 16 bits**. Read-only,Volatile. Reset: 0. 1=Port access was 16 bits.
4	**SZ8**: **size 8 bits**. Read-only,Volatile. Reset: 0. 1=Port access was 8 bits.
3	**REP**: **repeated port access**. Read-only,Volatile. Reset: 0.
2	**STR**: **string-based port access**. Read-only,Volatile. Reset: 0.
1	**V**: **IO trap word valid**. 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	**RW**: **port access type**. 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	**MceRedirSts**: **machine check exception redirection status**. 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	**IoTrapSts**: **IO trap status**. 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	**RST**: **SMM IO Restart Byte**. Read-write. Reset: 00h.

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

Read-write. Reset: 00h.

Bits	Description
7:1	Reserved.
0	**HLT**: **halt restart**. 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	**NmiMask**: **NMI Mask**. 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.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:5</td><td>Reserved.</td></tr>
<tr><td>4</td><td><b>SmmFromSev</b>. Read-only,Volatile. Reset: 0. 1= SMM was entered while executing on a guest with SEV enabled.</td></tr>
<tr><td>3</td><td><b>HostEflagesIF</b>: <b>host EFLAGS IF</b>. Read-only,Volatile. Reset: 0.</td></tr>
<tr><td rowspan="2">2:0</td><td><b>SvmState</b>. Read-only,Volatile. Reset: 0h.</td></tr>
<tr><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>SMM entered from a non-guest state.</td></tr>
<tr><td>1h</td><td>Reserved.</td></tr>
<tr><td>2h</td><td>SMM entered from a guest state.</td></tr>
<tr><td>5h-3h</td><td>Reserved.</td></tr>
<tr><td>6h</td><td>SMM entered from a guest state with nested paging enabled.</td></tr>
<tr><td>7h</td><td>Reserved.</td></tr>
</table></td></tr>
</table>

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	**BRL**. Read-only. Reset: 1. 1=Base relocation supported.
16	**IOTrap**. Read-only. Reset: 1. 1=IO trap supported.
15:0	**Revision**. 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	**SmmBase**. Read-write,Volatile. Reset: 0000_0000h. See Core::X86::Msr::SMM_BASE[SmmBase].

2.1.12.1.7 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.

2.1.12.1.8 The Protected ASeg and TSeg Areas

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

2.1.12.1.9 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].

2.1.12.1.10 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.

2.1.12.2 Local APIC

2.1.12.2.1 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.

2.1.12.2.1.1 Detecting and Enabling

APIC is detected and enabled via Core::X86::Cpuid::FeatureIdEdx[APIC].

The local APIC is enabled via Core::X86::Msr::APIC_BAR[ApicEn]. Reset forces APIC disabled.

2.1.12.2.1.2 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 APICXXX; XXX is the byte address offset from the base address.
- MMIO APIC registers memory in xAPIC mode is defined by the register starting with Core::X86::Apic::ApicId at offset (Core::X86::Msr::APIC_BAR[ApicBar[47:12]] + 20h) ending with Core::X86::Apic::ExtendedInterruptLvtEntries at offsets Core::X86::Msr::APIC_BAR[ApicBar[47:12]] + 533h.
- Treated as normal memory space when APIC is disabled, as specified by Core::X86::Msr::APIC_BAR[ApicEn].

2.1.12.2.1.3 ApicId Enumeration Requirements

Note: Family 17h processors 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].

Each Core::X86::Apic::ApicId[ApicId] register is preset as follows:

- ApicId[6] = Socket ID.
- ApicId[5:4] = Node ID.
- ApicId[3] = Logical CCX L3 complex ID
- ApicId[2:0]= (SMT) ? {LogicalCoreID[1:0],ThreadId} : {1'b0,LogicalCoreID[1:0]}.

2.1.12.2.1.4 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.

2.1.12.2.1.5 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.

2.1.12.2.1.6 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.

2.1.12.2.1.7 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.

2.1.12.2.1.8 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.

2.1.12.2.1.9 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.

2.1.12.2.1.10 Spurious Interrupts Caused by Timer Tick Interrupt

A typical interrupt is asserted until it is serviced. An interrupt is deasserted when software clears the interrupt status bit within the interrupt service routine. Timer tick interrupt is an exception, since it is deasserted 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 deasserted. The processor sends the interrupt acknowledge cycle, but when the PIC receives it, INTR is deasserted, 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.

2.1.12.2.1.11 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]
} elsif { (InterruptRequest[IRRVec[7:4]] > InService[ISRVec[7:4]])
ArbitrationPriority[Priority] = {InterruptRequest[IRRVec[7:4]],0h}
} else {
ArbitrationPriority[Priority] = {InService[ISRVect[7:4]],0h}
}
```

2.1.12.2.1.12 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.

Message type (bits[10:8]) == 011b (Remote Read) is deprecated.

Not all combinations of ICR fields are valid. Only the following combinations are valid:
Note: x indicates a don't care.

Table 18: 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

2.1.12.2.1.13 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.

2.1.12.2.1.14 Generalized Local Vector Table

All LVTs (Core::X86::Apic::ThermalLvtEntry to Core::X86::Apic::LVTLINT, 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.

2.1.12.2.1.15 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.

2.1.12.2.2 Local APIC Registers

APICx020 [**APIC ID**] **(Core::X86::Apic::ApicId)**	
Read-only.	
_lthree[1:0]_core[3:0]_thread[1:0]; APICx020; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}	
Bits	**Description**
31:24	**ApicId**: **APIC ID**. Read-only. Reset: XXh. The reset value varies based on core number. See 2.1.12.2.1.3 [ApicId Enumeration Requirements].
23:0	Reserved.

APICx030 [**APIC Version**] **(Core::X86::Apic::ApicVersion)**	
Read-only.	
_lthree[1:0]_core[3:0]_thread[1:0]; APICx030; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}	
Bits	**Description**
31	**ExtApicSpace**: **extended APIC register space present**. Read-only. Reset: 1. 1=Indicates the presence of extended APIC register space starting at Core::X86::Apic::ExtendedApicFeature.
30:25	Reserved.

24	**DirectedEoiSupport**: **directed EOI support**. Read-only. Reset: Fixed,0. 0=Directed EOI capability not supported.
23:16	**MaxLvtEntry**. Read-only. Reset: XXh. Specifies the number of entries in the local vector table minus one.
15:8	Reserved.
7:0	**Version**. Read-only. Reset: 10h. Indicates the version number of this APIC implementation.

APICx080 [Task Priority] (Core::X86::Apic::TaskPriority)

Read-write. Reset: 0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx080; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:8	Reserved.
7:0	**Priority**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx090; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:8	Reserved.
7:0	**Priority**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx0A0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:8	Reserved.
7:0	**Priority**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx0B0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	Reserved. Write-only.

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

Read-only. Reset: 0000_0000h.

Remote Read is deprecated.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx0C0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	Reserved. Read-only. Reset: 0000_0000h.

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

Read-write,Volatile. Reset: 0000_0000h.

_lthree[1:0]_core[3:0]_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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx0E0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:28</td><td><b>Format</b>. Read-write. Reset: Fh. Controls which format to use when accepting interrupts with a logical destination mode.
<b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Cluster destinations are used.</td></tr>
<tr><td>Eh-1h</td><td>Reserved.</td></tr>
<tr><td>Fh</td><td>Flat destinations are used.</td></tr>
</table></td></tr>
<tr><td>27:0</td><td>Reserved.</td></tr>
</table>

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

Reset: 0000_00FFh.

_lthree[1:0]_core[3:0]_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.

_lthree[1:0]_core[3:0]_thread[1:0]_n0; APICx100; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n1; APICx110; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n2; APICx120; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n3; APICx130; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n4; APICx140; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n5; APICx150; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n6; APICx160; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n7; APICx170; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	**InServiceBits**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_n0; APICx180; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n1; APICx190; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n2; APICx1A0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n3; APICx1B0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n4; APICx1C0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n5; APICx1D0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n6; APICx1E0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n7; APICx1F0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	**TriggerModeBits**. 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.

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.

_lthree[1:0]_core[3:0]_thread[1:0]_n0; APICx200; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n1; APICx210; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n2; APICx220; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n3; APICx230; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n4; APICx240; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n5; APICx250; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n6; APICx260; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n7; APICx270; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	**RequestBits**. Read-only. Reset: 0000_0000h. The corresponding request bit is set when the an interrupt is accepted by the APIC.

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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx280; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:8	Reserved.
7	**IllegalRegAddr**: **illegal register address**. 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	**RcvdIllegalVector**: **received illegal vector**. 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	**SentIllegalVector**. 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	**RcvAcceptError**: **receive accept error**. Read-write. Reset: 0. This bit indicates that a message received by this APIC was not accepted by this or any other APIC.
2	**SendAcceptError**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx300; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:20	Reserved.
19:18	**DestShrthnd**: **destination shorthand**. Read-write. Reset: 0h.
	Description: 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.
	ValidValues:

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	**RemoteRdStat**. Read-only. Reset: 0h.	
	ValidValues:	
	Value	**Description**
	0h	Read was invalid.
	1h	Delivery pending.
	2h	Delivery complete and access was valid.
	3h	Reserved.
15	**TM**: **trigger mode**. Read-write. Reset: 0. 0=Edge triggered. 1=Level triggered. Indicates how this interrupt is triggered.	
14	**Level**. Read-write. Reset: 0. 0=Deasserted. 1=Asserted.	
13	Reserved.	
12	**DS**: **interrupt delivery status**. 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	**DM**: **destination mode**. Read-write. Reset: 0. 0=Physical. 1=Logical.	
10:8	**MsgType**. Read-write. Reset: 0h. The message types are encoded as follows:	
	ValidValues:	
	Value	**Description**
	0h	Fixed
	1h	Lowest Priority.
	2h	SMI
	3h	Reserved.
	4h	NMI
	5h	INIT
	6h	Startup
	7h	External interrupt.
7:0	**Vector**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx310; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:24	**DestinationField**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx320; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:18	Reserved.
17	**Mode**. Read-write. Reset: 0. 0=One-shot. 1=Periodic.
16	**Mask**. Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	**DS**: **interrupt delivery status**. 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	**MsgType**: **message type**. Read-write. Reset: 0h. See2.1.12.2.1.14 [Generalized Local Vector Table].
7:0	**Vector**. Read-write. Reset: 00h. Interrupt vector number.

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

Reset: 0001_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx330; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	**Mask**. Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	**DS**: **interrupt delivery status**. 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	**MsgType**: **message type**. Read-write. Reset: 0h. See2.1.12.2.1.14 [Generalized Local Vector Table].
7:0	**Vector**. 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_CTL(Performance Event Select [3:0]).
- Core::X86::Msr::PERF_CTL(Performance Event Select [5:0]).

_lthree[1:0]_core[3:0]_thread[1:0]; APICx340; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	**Mask**. Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	**DS**: **interrupt delivery status**. 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	**MsgType**: **message type**. Read-write. Reset: 0h. See2.1.12.2.1.14 [Generalized Local Vector Table].
7:0	**Vector**. Read-write. Reset: 00h. Interrupt vector number.

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

Reset: 0001_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]_n0; APICx350; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

_lthree[1:0]_core[3:0]_thread[1:0]_n1; APICx360; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	**Mask**. Read-write. Reset: 1. 0=Not masked. 1=Masked.
15	**TM**: **trigger mode**. Read-write. Reset: 0. 0=Edge. 1=Level.
14	**RmtIRR**. 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. Read-write. Reset: 0.
12	**DS**: **interrupt delivery status**. 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	**MsgType**: **message type**. Read-write. Reset: 0h. See2.1.12.2.1.14 [Generalized Local Vector Table].
7:0	**Vector**. Read-write. Reset: 00h. Interrupt vector number.

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

Reset: 0001_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx370; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:17	Reserved.
16	**Mask**. Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	**DS**: **interrupt delivery status**. 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	**MsgType**: **message type**. Read-write. Reset: 0h. See 2.1.12.2.1.14 [Generalized Local Vector Table].
7:0	**Vector**. Read-write. Reset: 00h. Interrupt vector number.

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

Read-write,Volatile. Reset: 0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx380; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	**Count**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx390; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	**Count**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx3E0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:4	Reserved.
3:0	**Div[3:0]**. Read-write. Reset: 0h. Div[2] is unused.

ValidValues:

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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx400; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:24	Reserved.
23:16	**ExtLvtCount**: **extended local vector table count**. Read-only. Reset: 04h. This specifies the number of extended LVT registers (Core::X86::Apic::ExtendedInterruptLvtEntries) in the local APIC.

15:3	Reserved.
2	**ExtApicIdCap**: **extended APIC ID capable**. 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	**SeoiCap**: **specific end of interrupt capable**. Read-only. Reset: 1. 1=The Core::X86::Apic::SpecificEndOfInterrupt is present.
0	**IerCap**: **interrupt enable register capable**. Read-only. Reset: 1. This bit indicates that the Core::X86::Apic::InterruptEnable are present. See2.1.12.2.1.8 [Interrupt Masking].

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

Read-write. Reset: 0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx410; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:3	Reserved.
2	**ExtApicIdEn**: **extended APIC ID enable**. 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	**SeoiEn**. Read-write. Reset: 0. 1=Enable SEOI generation when a write to Core::X86::Apic::SpecificEndOfInterrupt is received.
0	**IerEn**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; APICx420; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:8	Reserved.
7:0	**EoiVec**: **end of interrupt vector**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_n0; APICx480; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n1; APICx490; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n2; APICx4A0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n3; APICx4B0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n4; APICx4C0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n5; APICx4D0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n6; APICx4E0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n7; APICx4F0; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

Bits	Description
31:0	**InterruptEnableBits**. 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].

_lthree[1:0]_core[3:0]_thread[1:0]_n0; APICx500; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n1; APICx510; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}
_lthree[1:0]_core[3:0]_thread[1:0]_n2; APICx520; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}

_lthree[1:0]_core[3:0]_thread[1:0]_n3; APICx530; APIC={Core::X86::Msr::APIC_BAR[ApicBar[47:12]] , 000h}	
Bits	**Description**
31:17	Reserved.
16	**Mask**. Read-write. Reset: 1. 0=Not masked. 1=Masked.
15:13	Reserved.
12	**DS**: **interrupt delivery status**. 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	**MsgType**: **message type**. Read-write. Reset: 0h. See2.1.12.2.1.14 [Generalized Local Vector Table].
7:0	**Vector**. Read-write. Reset: 00h. Interrupt vector number.

2.1.13 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 FnXXXX_XXXX_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.

2.1.13.1 CPUID Instruction Functions

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.

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_000Dh.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000000_EAX	
Bits	**Description**
31:0	**LFuncStd**: **largest standard function**. Read-only. Reset: Fixed,0000_000Dh. 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.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000000_EBX	
Bits	**Description**
31:0	**Vendor**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000000_ECX

Bits	Description
31:0	**Vendor**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000000_EDX

Bits	Description
31:0	**Vendor**. 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.

Model numbers are are assigned a letter, 0h = "A", 1h = "B", and so on. Model and Stepping form the Revision. E.g., B2.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000001_EAX

Bits	Description
31:28	Reserved.
27:20	**ExtFamily**: **extended family**. Read-only. Reset: 08h. See Family above.
19:16	**ExtModel**: **extended model**. Read-only. Reset: 0h. See Model above.
15:12	Reserved.
11:8	**BaseFamily**. Read-only. Reset: Fh. See Family description above.
7:4	**BaseModel**. Read-only. Reset: Xh. Model numbers vary with product.
3:0	**Stepping**. Read-only. Reset: 2h. Processor stepping (revision) for a specific model.

CPUID_Fn00000001_EBX [LocalApicId, LogicalProcessorCount, CLFlush] (Core::X86::Cpuid::FeatureIdEbx)

Read-only.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000001_EBX

Bits	Description
31:24	**LocalApicId**. Read-only. Reset: XXh. Initial local APIC physical ID.
23:16	**LogicalProcessorCount**: **logical processor count**. Read-only. Reset: Fixed,(Core::X86::Cpuid::SizeId[NC] + 1). Specifies the number of threads in the processor as Core::X86::Cpuid::SizeId[NC]+1.
15:8	**CLFlush**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000001_ECX

Bits	Description
31	Reserved. Read-only. Reset: Fixed,0. Reserved for use by hypervisor to indicate guest status.
30	**RDRAND**. Read-only. Reset: Fixed,1. RDRAND instruction support.

29	**F16C**. Read-only. Reset: Fixed,1. Half-precision convert instruction support.
28	**AVX**. Read-only. Reset: Fixed,1. AVX instruction support.
27	**OSXSAVE**. 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	**XSAVE**. 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	**AES**: **AES instruction support**. Read-only. Reset: X. AES instruction support.
24	Reserved.
23	**POPCNT**. Read-only. Reset: Fixed,1. POPCNT instruction.
22	**MOVBE**. Read-only. Reset: Fixed,1. MOVBE instruction support.
21	**X2APIC**. Read-only. Reset: Fixed,0. x2APIC capability.
20	**SSE42**. Read-only. Reset: Fixed,1. SSE4.2 instruction support.
19	**SSE41**. Read-only. Reset: Fixed,1. SSE4.1 instruction support.
18	Reserved.
17	**PCID**. Read-only. Reset: Fixed,0. Process context identifiers support.
16:14	Reserved.
13	**CMPXCHG16B**. Read-only. Reset: Fixed,1. CMPXCHG16B instruction.
12	**FMA**. Read-only. Reset: Fixed,1. FMA instruction support.
11:10	Reserved.
9	**SSSE3**. Read-only. Reset: Fixed,1. Supplemental SSE3 extensions.
8:4	Reserved.
3	**Monitor**. Read-only. Reset: !Core::X86::Msr::HWCR[MonMwaitDis]. Monitor/Mwait instructions.
2	Reserved.
1	**PCLMULQDQ**. Read-only. Reset: X. PCLMULQDQ instruction support.
0	**SSE3**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000001_EDX

Bits	Description
31:29	Reserved.
28	**HTT**. 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	**SSE2**. Read-only. Reset: Fixed,1. SSE2: SSE2 extensions.
25	**SSE**. Read-only. Reset: Fixed,1. SSE extensions.
24	**FXSR**. Read-only. Reset: Fixed,1. FXSAVE and FXRSTOR instructions.
23	**MMX**. Read-only. Reset: Fixed,1. MMX™ instructions
22:20	Reserved.
19	**CLFSH**. Read-only. Reset: Fixed,1. CLFLUSH instruction.
18	Reserved.
17	**PSE36**. Read-only. Reset: Fixed,1. Page-size extensions.
16	**PAT**. Read-only. Reset: Fixed,1. Page attribute table.
15	**CMOV**. Read-only. Reset: Fixed,1. Conditional move instructions, CMOV, FCOMI, FCMOV.
14	**MCA**. Read-only. Reset: Fixed,1. Machine check architecture, MCG_CAP.
13	**PGE**. Read-only. Reset: Fixed,1. Page global extension, CR4.PGE.
12	**MTRR**. Read-only. Reset: Fixed,1. Memory-type range registers.

11	**SysEnterSysExit**. Read-only. Reset: Fixed,1. SYSENTER and SYSEXIT instructions.
10	Reserved.
9	**APIC**: **advanced programmable interrupt controller (APIC) exists and is enabled**. Read-only. Reset: X. Core::X86::Msr::APIC_BAR[ApicEn].
8	**CMPXCHG8B**. Read-only. Reset: Fixed,1. CMPXCHG8B instruction.
7	**MCE**. Read-only. Reset: Fixed,1. Machine check exception, CR4.MCE.
6	**PAE**. Read-only. Reset: Fixed,1. physical-address extensions (PAE).
5	**MSR**. Read-only. Reset: Fixed,1. AMD model-specific registers (MSRs), with RDMSR and WRMSR instructions.
4	**TSC**. Read-only. Reset: Fixed,1. Time Stamp Counter, RDTSC/RDTSCP instructions, CR4.TSD.
3	**PSE**. Read-only. Reset: Fixed,1. Page-size extensions (4 MB pages).
2	**DE**. Read-only. Reset: Fixed,1. Debugging extensions, IO breakpoints, CR4.DE.
1	**VME**. Read-only. Reset: Fixed,1. Virtual-mode enhancements.
0	**FPU**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000005_EAX

Bits	Description
31:16	Reserved.
15:0	**MonLineSizeMin**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000005_EBX

Bits	Description
31:16	Reserved.
15:0	**MonLineSizeMax**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000005_ECX

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

CPUID_Fn00000005_EDX [Monitor/MWait] (Core::X86::Cpuid::MonMWaitEdx)

Read-only. Reset: Fixed,0000_0011h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000005_EDX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000006_EAX

Bits	Description
31:3	Reserved.
2	**ARAT**: **always running APIC timer**. 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)

_lthree[1:0]_core[3:0]_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::Msr::CPUID_PWR_THERM.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000006_ECX

Bits	Description
31:1	Reserved.
0	**EffFreq**: **effective frequency interface**. Read-only. Reset: Fixed,1. 1=Indicates presence of Core::X86::Msr::MPERF and Core::X86::Msr::APERF.

CPUID_Fn00000006_EDX [Thermal and Power Management] (Core::X86::Cpuid::ThermalPwrMgmtEdx)

_lthree[1:0]_core[3:0]_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_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000007_EAX_x00

Bits	Description
31:0	**StructExtFeatIdMax**. Read-only. Reset: Fixed,0000_0000h. 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,209C_01A9h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000007_EBX_x00

Bits	Description
31:30	Reserved.
29	**SHA**. Read-only. Reset: Fixed,1.
28:24	Reserved.
23	**CLFSHOPT**. Read-only. Reset: Fixed,1. Optimized Cache Line Flush.
22:21	Reserved.
20	**SMAP**. Read-only. Reset: Fixed,1. Secure Mode Access Prevention is supported.
19	**ADX**. Read-only. Reset: Fixed,1. ADCX and ADOX are present.
18	**RDSEED**. Read-only. Reset: Fixed,1. RDSEED is present.
17:9	Reserved.
8	**BMI2**. Read-only. Reset: Fixed,1. Bit manipulation group 2 instruction support.
7	**SMEP**. Read-only. Reset: Fixed,1. Supervisor Mode Execution protection.
6	Reserved.
5	**AVX2**. Read-only. Reset: Fixed,1. AVX extension support.
4	Reserved.
3	**BMI1**. Read-only. Reset: Fixed,1. Bit manipulation group 1 instruction support.
2:1	Reserved.
0	**FSGSBASE**. Read-only. Reset: Fixed,1. FS and GS base read write instruction support.

CPUID_Fn00000007_ECX_x00 [Structured Extended Feature Identifier] (Core::X86::Cpuid::StructExtFeatIdEcx0)

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000007_ECX_x00

Bits	Description
31:0	Reserved.

CPUID_Fn00000007_EDX_x00 [Structured Extended Feature Identifiers] (Core::X86::Cpuid::StructExtFeatIdEdx0)

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn00000007_EDX_x00

Bits	Description
31:0	Reserved.

CPUID_Fn0000000B_EAX [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEax)

Read-only. Reset: Fixed,0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000B_EAX

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn0000000B_EBX [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEbx)

Read-only. Reset: Fixed,0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000B_EBX

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn0000000B_ECX [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEcx)

Read-only. Reset: Fixed,0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000B_ECX

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn0000000B_EDX [Extended Topology Enumeration] (Core::X86::Cpuid::ExtTopEnumEdx)

Read-only. Reset: Fixed,0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000B_EDX

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn0000000D_EAX_x00 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEax00)

Read-only. Reset: Fixed,0000_0007h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EAX_x00

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:0</td><td><b>XFeatureSupportedMask[31:0]</b>. Read-only. Reset: Fixed,0000_0007h. Each set bit indicates the corresponding bit in register XCR0[31:0] is settable.
<b>ValidValues</b>:
<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>[0]</td><td>X87</td><td>X87 Support.</td></tr>
<tr><td>[1]</td><td>SSE</td><td>128-bit SSE Support.</td></tr>
<tr><td>[2]</td><td>AVX</td><td>256-bit AVX support.</td></tr>
<tr><td>[31:3]</td><td></td><td>Reserved.</td></tr>
</table>
</td></tr>
</table>

CPUID_Fn0000000D_EBX_x00 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEbx00)

Read-only,Volatile.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EBX_x00	
Bits	**Description**
31:0	**XFeatureEnabledSizeMax**. Read-only,Volatile. Reset: XXXX_XXXXh.

	Description: Size in bytes of an uncompacted XSAVE/XRSTOR area for all features enabled in the XCR0 register.
	IF (XCR0[MPK]==1)
	    return EBX=0000_0360h // legacy header + X87/SSE + AVX + MPK size
	ELSIF (XCR0[AVX]==1)
	    return EBX=0000_0340h // legacy header + X87/SSE + AVX size
	ELSE
	    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_0360h.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_ECX_x00	
Bits	**Description**
31:0	**XFeatureSupportedSizeMax**. Read-only. Reset: Fixed,0000_0360h. Size of legacy header + X87/SSE + AVX + MPK.

CPUID_Fn0000000D_EDX_x00 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEdx00)

Read-only. Reset: Fixed,0000_0000h.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EDX_x00	
Bits	**Description**
31:0	**XFeatureSupportedMask[63:32]**. 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.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EAX_x01	
Bits	**Description**
31:4	Reserved.
3	**XSAVES**. Read-only. Reset: Fixed,1. XSAVES,XRSTORS, and IA32_XSS supported.
2	**XGETBV**. Read-only. Reset: Fixed,1. XGETBV with ECX=1 supported.
1	**XSAVEC**. Read-only. Reset: Fixed,1. XSAVEC and compact XRSTOR supported.
0	**XSAVEOPT**. Read-only. Reset: Fixed,1. XSAVEOPT is available.

CPUID_Fn0000000D_EBX_x01 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEbx01)

Read-only,Volatile.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EBX_x01	
Bits	**Description**
31:0	**XFeatureEnabledSizeMax**. Read-only,Volatile. Reset: XXXX_XXXXh. Reset is 512 + ((XCR0[AVX]) ? 256 : 0).
	ValidValues:

Value	**Description**
0000_023Fh-	Reserved.

	0000_0000h	
	0000_0240h	legacy header + FPU/SSE size; (XCR0[AVX]==0)
	0000_033Fh-0000_0241h	Reserved.
	0000_0340h	legacy header + FPU/SSE + AVX size; (XCR0[AVX]==1)
	FFFF_FFFFh-0000_0341h	Reserved.

CPUID_Fn0000000D_ECX_x01 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEcx01)

Read-only. Reset: Fixed,0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_ECX_x01

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn0000000D_EDX_x01 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEdx01)

Read-only. Reset: Fixed,0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EDX_x01

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn0000000D_EAX_x02 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEax02)

Read-only. Reset: Fixed,0000_0100h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EAX_x02

Bits	Description
31:0	**YmmSaveStateSize**. Read-only. Reset: Fixed,0000_0100h. YMM save state byte size.

CPUID_Fn0000000D_EBX_x02 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEbx02)

Read-only. Reset: Fixed,0000_0240h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EBX_x02

Bits	Description
31:0	**YmmSaveStateOffset**. Read-only. Reset: Fixed,0000_0240h. YMM save state byte offset.

CPUID_Fn0000000D_ECX_x02 [Processor Extended State Enumeration] (Core::X86::Cpuid::ProcExtStateEnumEcx02)

Read-only. Reset: Fixed,0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_ECX_x02

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn0000000D_EDX_x02 [Processor Extended State Enumeration]

(Core::X86::Cpuid::ProcExtStateEnumEdx02)

Read-only. Reset: Fixed,0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn0000000D_EDX_x02

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn80000000_EAX [Largest Extended Function Number] (Core::X86::Cpuid::LargExtFuncNum)

Read-only. Reset: Fixed,8000_001Fh.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000000_EAX

Bits	Description
31:0	**LFuncExt**: **largest extended function**. Read-only. Reset: Fixed,8000_001Fh. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000000_EBX

Bits	Description
31:0	**Vendor**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000000_ECX

Bits	Description
31:0	**Vendor**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000000_EDX

Bits	Description
31:0	**Vendor**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000001_EAX

Bits	Description
31:28	Reserved.
27:20	**ExtFamily**: **extended family**. Read-only. Reset: 08h. See Core::X86::Cpuid::FamModStep description of Family.
19:16	**ExtModel**: **extended model**. Read-only. Reset: 0h. See Core::X86::Cpuid::FamModStep description of ExtModel.
15:12	Reserved.
11:8	**BaseFamily**. Read-only. Reset: Fh. See Core::X86::Cpuid::FamModStep description of Family.
7:4	**BaseModel**. Read-only. Reset: Xh. Model numbers vary with product.
3:0	**Stepping**. Read-only. Reset: 2h. Processor stepping (revision) for a specific model.

CPUID_Fn80000001_EBX [BrandId Identifier] (Core::X86::Cpuid::BrandId)

Read-only.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000001_EBX

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:28</td><td><b>PkgType</b>: <b>package type</b>. Read-only. Reset: Xh. Specifies the package type.</td></tr>
<tr><td></td><td><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>AM4</td></tr>
<tr><td>Fh-3h</td><td>Reserved.</td></tr>
</table></td></tr>
<tr><td>27:0</td><td>Reserved.</td></tr>
</table>

CPUID_Fn80000001_ECX [Feature Identifiers] (Core::X86::Cpuid::FeatureExtIdEcx)

Read-only.

These values can be over-written by Core::X86::Msr::CPUID_ExtFeatures.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000001_ECX

Bits	Description
31:30	Reserved.
29	**MwaitExtended**. Read-only. Reset: !Core::X86::Msr::HWCR[MonMwaitDis]. 1=MWAITX and MONITORX capability is supported.
28	**PerfCtrExtLLC**: **Last Level Cache performance counter extensions**. Read-only. Reset: Fixed,1. 1=Indicates support for Core::X86::Msr::ChL3PmcCfg and Core::X86::Msr::ChL3Pmc L3 performance counter extensions. L3 performance counter extensions support. See 2.1.15.5 [L3 Cache Performance Monitor Counters] and 2.1.15 [Performance Monitor Counters].
27	**PerfTsc**. Read-only. Reset: Fixed,0. Performance time-stamp counter supported.
26	**DataBreakpointExtension**. 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.
25:24	Reserved.
23	**PerfCtrExtCore**: **core performance counter extensions support**. Read-only. Reset: Fixed,1. 1=Indicates support for Core::X86::Msr::PERF_CTL and Core::X86::Msr::PERF_CTR. See See 2.1.15.4 [Core Performance Monitor Counters] and 2.1.15 [Performance Monitor Counters].
22	**TopologyExtensions**: **topology extensions support**. Read-only. Reset: Fixed,1. 1=Indicates support for Core::X86::Cpuid::CachePropEax0 and Core::X86::Cpuid::ExtApicId.
21:18	Reserved.
17	**TCE**. Read-only. Reset: Fixed,1. Translation cache extension.
16	**FMA4**. Read-only. Reset: Fixed,0. Four-operand FMA instruction support.
15	**LWP**. Read-only. Reset: Fixed,0. Lightweight profiling support.
14	Reserved.
13	**WDT**. Read-only. Reset: Fixed,1. Watchdog timer support.
12	**SKINIT**. Read-only. Reset: Fixed,1. SKINIT and STGI support.
11	**XOP**. Read-only. Reset: Fixed,0. Extended operation support.
10	**IBS**. Read-only. Reset: 0. Instruction Based Sampling.
9	**OSVW**. Read-only. Reset: Fixed,1. OS Visible Work-around support.
8	**ThreeDNowPrefetch**. Read-only. Reset: Fixed,1. Prefetch and PrefetchW instructions.
7	**MisAlignSse**. Read-only. Reset: Fixed,1. Misaligned SSE Mode.
6	**SSE4A**. Read-only. Reset: Fixed,1. EXTRQ, INSERTQ, MOVNTSS, and MOVNTSD instruction support.
5	**ABM**: **advanced bit manipulation**. Read-only. Reset: Fixed,1. LZCNT instruction support.
4	**AltMovCr8**. Read-only. Reset: Fixed,1. LOCK MOV CR0 means MOV CR8.
3	**ExtApicSpace**. Read-only. Reset: Fixed,1. Extended APIC register space.
2	**SVM**: **Secure Virtual Mode feature**. Read-only. Reset: Fixed,1. Indicates support for: VMRUN, VMLOAD, VMSAVE, CLGI, VMMCALL, and INVLPGA.
1	**CmpLegacy**. 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	**LahfSahf**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000001_EDX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000002_EAX

Bits	Description
31:24	**ProcNameByte3**. Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString3]. Processor name, byte3.

23:16	**ProcNameByte2**. Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString2]. Processor name, byte2.
15:8	**ProcNameByte1**. Read-only. Reset: Core::X86::Msr::ProcNameString_n0[CpuNameString1]. Processor name, byte1.
7:0	**ProcNameByte0**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000002_EBX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000002_ECX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000002_EDX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000003_EAX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000003_EBX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000003_ECX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000003_EDX

Bits	Description
31:24	**ProcNameByte31**. Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString7]. Processor name,

	byte 31.
23:16	**ProcNameByte30**. Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString6]. Processor name, byte 30.
15:8	**ProcNameByte29**. Read-only. Reset: Core::X86::Msr::ProcNameString_n3[CpuNameString5]. Processor name, byte 29.
7:0	**ProcNameByte28**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000004_EAX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000004_EBX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000004_ECX

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

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000004_EDX

Bits	Description
31:24	**ProcNameByte47**. Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString7]. Processor name, byte 47.
23:16	**ProcNameByte46**. Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString6]. Processor name, byte 46.
15:8	**ProcNameByte45**. Read-only. Reset: Core::X86::Msr::ProcNameString_n5[CpuNameString5]. Processor name, byte 45.
7:0	**ProcNameByte44**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000005_EAX

Bits	Description
31:24	**L1DTlb2and4MAssoc**: **data TLB associativity for 2 MB and 4 MB pages**. Read-only. Reset: Fixed,FFh. See Core::X86::Cpuid::L1DcId[L1DcAssoc].
23:16	**L1DTlb2and4MSize**: **data TLB number of entries for 2 MB and 4 MB pages**. 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.
15:8	**L1ITlb2and4MAssoc**: **instruction TLB associativity for 2 MB and 4 MB pages**. Read-only. Reset: Fixed,FFh. See Core::X86::Cpuid::L1DcId[L1DcAssoc].
7:0	**L1ITlb2and4MSize**: **instruction TLB number of entries for 2 MB and 4 MB pages**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000005_EBX

Bits	Description
31:24	**L1DTlb4KAssoc**. Read-only. Reset: Fixed,FFh. Data TLB associativity for 4 KB pages. See Core::X86::Cpuid::L1DcId[L1DcAssoc].
23:16	**L1DTlb4KSize**. Read-only. Reset: Fixed,64. Data TLB number of entries for 4 KB pages.
15:8	**L1ITlb4KAssoc**. Read-only. Reset: Fixed,FFh. Instruction TLB associativity for 4 KB pages. See Core::X86::Cpuid::L1DcId[L1DcAssoc].
7:0	**L1ITlb4KSize**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000005_ECX

Bits	Description
31:24	**L1DcSize**. Read-only. Reset: Fixed,32. L1 data cache size in KB.
23:16	**L1DcAssoc**. Read-only. Reset: Fixed,8. L1 data cache associativity.
	ValidValues:

Value	Description
00h	Reserved
01h	1 way (direct mapped)
02h	2 way
03h	3 way
FEh-04h	<Value> way
FFh	Fully associative

15:8	**L1DcLinesPerTag**. Read-only. Reset: Fixed,01h. L1 data cache lines per tag.
7:0	**L1DcLineSize**. Read-only. Reset: Fixed,64. L1 data cache line size in bytes.

CPUID_Fn80000005_EDX [L1 Instruction Cache Identifiers] (Core::X86::Cpuid::L1IcId)

Read-only.

This function provides first level cache characteristics for each core.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000005_EDX

Bits	Description
31:24	**L1IcSize**. Read-only. Reset: Fixed,64. L1 instruction cache size KB.
23:16	**L1IcAssoc**. Read-only. Reset: Fixed,4. L1 instruction cache associativity.

ValidValues:

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

Bits	Description
15:8	**L1IcLinesPerTag**. Read-only. Reset: Fixed,01h. L1 instruction cache lines per tag.
7:0	**L1IcLineSize**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000006_EAX

Bits	Description
31:28	**L2DTlb2and4MAssoc**: **L2 data TLB associativity for 2 MB and 4 MB pages**. Read-only. Reset: Xh.

ValidValues:

Value	Description
1h-0h	Reserved.
2h	2 ways
3h	3 ways
Fh-4h	Reserved.

Bits	Description
27:16	**L2DTlb2and4MSize**: **L2 data TLB number of entries for 2 MB and 4 MB pages**. Read-only. Reset: Fixed,1536. 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.
15:12	**L2ITlb2and4MAssoc**: **L2 instruction TLB associativity for 2 MB and 4 MB pages**. Read-only. Reset: Fixed,6.

ValidValues:

Value	Description
5h-0h	Reserved.

<table>
<tr><td></td><td><table><tr><td>6h</td><td>8 ways</td></tr><tr><td>Fh-7h</td><td>Reserved.</td></tr></table></td></tr>
<tr><td>11:0</td><td><b>L2ITlb2and4MSize</b>: <b>L2 instruction TLB number of entries for 2 MB and 4 MB pages</b>. Read-only. Reset: Fixed,1024. 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.</td></tr>
</table>

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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000006_EBX

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:28</td><td><b>L2DTlb4KAssoc</b>. Read-only. Reset: Xh. L2 data TLB associativity for 4 KB pages.
<b>ValidValues</b>:<table><tr><th>Value</th><th>Description</th></tr><tr><td>4h-0h</td><td>Reserved.</td></tr><tr><td>5h</td><td>6 ways</td></tr><tr><td>6h</td><td>8 ways</td></tr><tr><td>Fh-7h</td><td>Reserved.</td></tr></table></td></tr>
<tr><td>27:16</td><td><b>L2DTlb4KSize</b>. Read-only. Reset: Fixed,1536. L2 data TLB number of entries for 4 KB pages.</td></tr>
<tr><td>15:12</td><td><b>L2ITlb4KAssoc</b>. Read-only. Reset: Fixed,6. L2 instruction 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 ways</td></tr><tr><td>Fh-7h</td><td>Reserved.</td></tr></table></td></tr>
<tr><td>11:0</td><td><b>L2ITlb4KSize</b>. Read-only. Reset: Fixed,1024. L2 instruction TLB number of entries for 4 KB pages.</td></tr>
</table>

CPUID_Fn80000006_ECX [L2 Cache Identifiers] (Core::X86::Cpuid::L2CacheId)

Read-only.

This function provides second level cache characteristics for each core.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000006_ECX

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td><b>L2Size</b>. Read-only. Reset: Fixed,0200h. 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></td></tr>
<tr><td>15:12</td><td><b>L2Assoc</b>. Read-only. Reset: Fixed,6. L2 cache associativity.</td></tr>
</table>

ValidValues:

Value	Description
0h	Disabled.
1h	1 way (direct mapped)
2h	2 ways
3h	Reserved.
4h	4 ways
5h	Reserved.
6h	8 ways
7h	Reserved.
8h	16 ways
9h	Reserved.
Ah	32 ways
Bh	48 ways
Ch	64 ways
Dh	96 ways
Eh	128 ways
Fh	Fully associative

11:8	**L2LinesPerTag**. Read-only. Reset: Fixed,1h. L2 cache lines per tag.
7:0	**L2LineSize**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000006_EDX

Bits	Description
31:18	**L3Size**: **L3 cache size**. Read-only. Reset: XXXXh. The L3 cache size in 512 KB units.

ValidValues:

Value	Description
0000h	Disabled.
3FFFh-0001h	(<Value> *0.5) MB

Bits	Description
17:16	Reserved.
15:12	**L3Assoc**. Read-only. Reset: Xh. L3 cache associativity.

ValidValues:

Value	Description
7h-0h	Reserved.
8h	16 Ways
9h	Reserved.
Ah	32 Ways
Bh	48 Ways
Ch	64 Ways
Dh	96 Ways
Eh	128 Ways
Fh	Reserved.

Bits	Description
11:8	**L3LinesPerTag**. Read-only. Reset: Fixed,1h. L3 cache lines per tag.
7:0	**L3LineSize**. 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.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000007_EAX	
Bits	**Description**
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn80000007_EBX [RAS Capabilities] (Core::X86::Cpuid::RasCap)

Read-only.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000007_EBX	
Bits	**Description**
31:4	Reserved.
3	**ScalableMca**. Read-only. Reset: Fixed,1. 0=Scalable MCA is not supported. 1=Scalable MCA is supported. See 3.1.1.2 [Machine Check Architecture Extensions] and MCA_CONFIG[McaX] for the respective bank.
2	**HWA**. Read-only. Reset: Fixed,0. Hardware assert supported.
1	**SUCCOR**: **Software uncorrectable error containment and recovery capability**. 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	**McaOverflowRecov**: **MCA overflow recovery support**. Read-only. Reset: Fixed,1. 0=MCA overflow conditions require software to shut down the system. 1=MCA overflow conditions (MCi_STATUS[Overflow] == 1) are not fatal; software may safely ignore such conditions. See 3.1 [Machine Check Architecture].

CPUID_Fn80000007_ECX [Advanced Power Management Information] (Core::X86::Cpuid::ApmInfoEcx)

Read-only. Reset: Fixed,0000_0000h.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000007_ECX	
Bits	**Description**
31:0	**CpuPwrSampleTimeRatio**. 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.	
_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000007_EDX	
Bits	**Description**
31:15	Reserved.
14	**RAPL**. Read-only. Reset: Fixed,1. Running average power limit.
13	**ConnectedStandby**. Read-only. Reset: Fixed,1. Connected Standby.
12	**ProcPowerReporting**. Read-only. Reset: Fixed,0. Core power reporting interface supported.
11	**ProcFeedbackInterface**: **processor feedback interface**. Read-only. Reset: Fixed,0. 1=Indicates support for processor feedback interface; Core::X86::Cpuid::ProcFeedbackCap.
10	**EffFreqRO**: **read-only effective frequency interface**. Read-only. Reset: Fixed,1. Indicates presence of Core::X86::Msr::MPerfReadOnly and Core::X86::Msr::APerfReadOnly.
9	**CPB**: **core performance boost**. Read-only. Reset: 0. 1=Indicates presence of Core::X86::Msr::HWCR[CpbDis] and support for core performance boost.
8	**TscInvariant**: **TSC invariant**. Read-only. Reset: Fixed,1. The TSC rate is invariant.
7	**HwPstate**: **hardware P-state control**. Read-only. Reset: Fixed,1. Core::X86::Msr::PStateCurLim, Core::X86::Msr::PStateCtl and Core::X86::Msr::PStateStat exist.
6	**OneHundredMHzSteps**. Read-only. Reset: Fixed,0. : 100 MHz multiplier Control.
5	Reserved.
4	**TM**. Read-only. Reset: Fixed,1. Hardware thermal control (HTC)
3	**TTP**. Read-only. Reset: Fixed,1. THERMTRIP.
2	**VID**: **Voltage ID control**. Read-only. Reset: Fixed,0. Function replaced by HwPstate.

1	**FID**: **Frequency ID control**. Read-only. Reset: Fixed,0. Function replaced by HwPstate.
0	**TS**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000008_EAX

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td>Reserved.</td></tr>
<tr><td rowspan="2">23:16</td><td><b>GuestPhysAddrSize</b>. Read-only. Reset: Fixed,00h. Maximum guest physical byte address size in bits.</td></tr>
<tr><td><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></td></tr>
<tr><td>15:8</td><td><b>LinAddrSize</b>. Read-only. Reset: Fixed,30h. Maximum linear byte address size in bits.</td></tr>
<tr><td>7:0</td><td><b>PhysAddrSize</b>. Read-only. Reset: Fixed,30h. Maximum physical byte address size in bits.</td></tr>
</table>

CPUID_Fn80000008_EBX [Extended Feature Extensions ID EBX] (Core::X86::Cpuid::FeatureExtIdEbx)

Read-only. Reset: Fixed,0000_0007h.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000008_EBX

Bits	Description
31:3	Reserved.
2	**RstrFpErrPtrs**. 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	**InstRetCntMsr**: **instructions retired count support**. Read-only. Reset: Fixed,1. 1=Core::X86::Msr::IRPerfCount supported.
0	**CLZERO**: **Clear Zero Instruction**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000008_ECX

Bits	Description
31:18	Reserved.
17:16	**PerfTscSize**: **performance time-stamp counter size**. Read-only. Reset: Fixed,0h.
15:12	**ApicIdSize**: **APIC ID size**. Read-only. Reset: Xh. The number of bits in the initial Core::X86::Apic::ApicId[ApicId] value that indicate thread ID within a package.
11:8	Reserved.
7:0	**NC**: **number of threads - 1**. Read-only. Reset: XXh. The number of threads in the package is NC+1 (eg., if NC=0, then there is one thread).

CPUID_Fn8000000A_EAX [SVM Revision and Feature Identification] (Core::X86::Cpuid::SvmRevFeatIdEax)

Read-only. Reset: Fixed,0000_0001h.

This provides SVM revision. If (Core::X86::Cpuid::FeatureExtIdEcx[SVM] == 0) then Core::X86::Cpuid::SvmRevFeatIdEax is reserved.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000000A_EAX

Bits	Description
31:8	Reserved.
7:0	**SvmRev**. 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. If (Core::X86::Cpuid::FeatureExtIdEcx[SVM] == 0) then Core::X86::Cpuid::SvmRevFeatIdEbx is reserved.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000000A_EBX

Bits	Description
31:0	**NASID**: **number of address space identifiers (ASID)**. Read-only,Volatile. Reset: 0000_8000h.

CPUID_Fn8000000A_EDX [SVM Revision and Feature Identification] (Core::X86::Cpuid::SvmRevFeatIdEdx)

Read-only. Reset: Fixed,0001_B4FFh.

This provides SVM feature information. If (Core::X86::Cpuid::FeatureExtIdEcx[SVM] == 0) then Core::X86::Cpuid::SvmRevFeatIdEdx is reserved.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000000A_EDX

Bits	Description
31:17	Reserved.
16	**vGIF**. Read-only. Reset: Fixed,1. Virtualized GIF.
15	**V_VMSAVE_VMLOAD**. Read-only. Reset: Fixed,1. Virtualized VMLOAD and VMSAVE.
14	Reserved.
13	**AVIC**: **AMD virtual interrupt controller**. Read-only. Reset: Fixed,1. 1=Support indicated for SVM mode virtualized interrupt controller; Indicates support for Core::X86::Msr::AvicDoorbell.
12	**PauseFilterThreshold**. Read-only. Reset: Fixed,1. PAUSE filter threshold.
11	Reserved.
10	**PauseFilter**. Read-only. Reset: Fixed,1. Pause intercept filter.
9:8	Reserved.
7	**DecodeAssists**. Read-only. Reset: Fixed,1. Decode assists.
6	**FlushByAsid**. Read-only. Reset: Fixed,1. Flush by ASID.
5	**VmcbClean**. Read-only. Reset: Fixed,1. VMCB clean bits.
4	**TscRateMsr**: **MSR based TSC rate control**. Read-only. Reset: Fixed,1. 1=Indicates support for TSC ratio Core::X86::Msr::TscRateMsr.
3	**NRIPS**. Read-only. Reset: Fixed,1. NRIP Save.
2	**SVML**. Read-only. Reset: Fixed,1. SVM lock.
1	**LbrVirt**. Read-only. Reset: Fixed,1. LBR virtualization.
0	**NP**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000019_EAX

Bits	Description
31:28	**L1DTlb1GAssoc**: **L1 data TLB associativity for 1 GB pages**. Read-only. Reset: Fixed,Fh. See Core::X86::Cpuid::L2CacheId[L2Assoc].
27:16	**L1DTlb1GSize**. Read-only. Reset: Fixed,64. L1 data TLB number of entries for 1 GB pages.
15:12	**L1ITlb1GAssoc**. Read-only. Reset: Fixed,Fh. L1 instruction TLB associativity for 1 GB pages. See Core::X86::Cpuid::L2CacheId[L2Assoc].
11:0	**L1ITlb1GSize**. 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. Reset: Fixed,0000_0000h.

This provides 1 GB paging information. The associativity fields are defined by Core::X86::Cpuid::L2Tlb2M4M, Core::X86::Cpuid::L2Tlb4K, Core::X86::Cpuid::L2CacheId and Core::X86::Cpuid::L3CacheId.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn80000019_EBX

Bits	Description
31:28	**L2DTlb1GAssoc**. Read-only. Reset: Fixed,0h. L2 data TLB associativity for 1 GB pages. See Core::X86::Cpuid::L2CacheId[L2Assoc].
27:16	**L2DTlb1GSize**. Read-only. Reset: Fixed,000h. L2 data TLB number of entries for 1 GB pages.
15:12	**L2ITlb1GAssoc**. Read-only. Reset: Fixed,0h. L2 instruction TLB associativity for 1 GB pages. See Core::X86::Cpuid::L2CacheId[L2Assoc].
11:0	**L2ITlb1GSize**. 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_0003h.

This function returns performance related information.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001A_EAX

Bits	Description
31:3	Reserved.
2	**FP256**. Read-only. Reset: Fixed,0. 256-bit AVX instructions are executed with full-width internal operations and pipelines rather than decomposing them into internal 128-bit suboperations.
1	**MOVU**. 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 MOVLPD/MOVHPD.
0	**FP128**. Read-only. Reset: Fixed,1. 128-bit SSE (multimedia) instructions are executed with full-width internal operations and pipelines rather than decomposing them into internal 64-bit suboperations.

CPUID_Fn8000001B_EAX [Instruction Based Sampling Identifiers] (Core::X86::Cpuid::IbsIdEax)

Read-only. Reset: Fixed,0000_03FFh.

This function returns IBS feature information.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001B_EAX

Bits	Description
31:11	Reserved.
10	**IbsOpData4**. Read-only. Reset: Fixed,0. IBS op data 4 MSR supported.
9	**IbsFetchCtlExtd**: **IBS fetch control extended MSR supported**. Read-only. Reset: Fixed,1. Indicates support for Core::X86::Msr::IC_IBS_EXTD_CTL.
8	**OpBrnFuse**: **fused branch micro-op indication supported**. Read-only. Reset: Fixed,1. Indicates support for Core::X86::Msr::IBS_OP_DATA[IbsOpBrnFuse].
7	**RipInvalidChk**: **invalid RIP indication supported**. Read-only. Reset: Fixed,1. Indicates support for Core::X86::Msr::IBS_OP_DATA[IbsRipInvalid].
6	**OpCntExt**: **IbsOpCurCnt and IbsOpMaxCnt extend by 7 bits**. Read-only. Reset: Fixed,1. Indicates support for Core::X86::Msr::IBS_OP_CTL[IbsOpCurCnt[26:20],IbsOpMaxCnt[26:20]].
5	**BrnTrgt**. Read-only. Reset: Fixed,1. Branch target address reporting supported.
4	**OpCnt**. Read-only. Reset: Fixed,1. Op counting mode supported.
3	**RdWrOpCnt**. Read-only. Reset: Fixed,1. Read write of op counter supported.
2	**OpSam**. Read-only. Reset: Fixed,1. IBS execution sampling supported.
1	**FetchSam**. Read-only. Reset: Fixed,1. IBS fetch sampling supported.
0	**IBSFFV**. 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.
If (Core::X86::Cpuid::FeatureExtIdEcx[TopologyExtensions] == 0) then CPUID Fn8000001D_E[D,C,B,A]X are reserved.

_lthree[1:0]_core[3:0]; CPUID_Fn8000001D_EAX_x00

Bits	Description
31:26	Reserved.

<table>
<tr><td>25:14</td><td><b>NumSharingCache</b>: <b>number of logical processors sharing cache</b>. Read-only. Reset: XXXh. The number of logical processors sharing this cache is NumSharingCache+1.</td></tr>
<tr><td>13:10</td><td>Reserved.</td></tr>
<tr><td>9</td><td><b>FullyAssociative</b>: <b>fully associative cache</b>. Read-only. Reset: Fixed,0. 1=Cache is fully associative.</td></tr>
<tr><td>8</td><td><b>SelfInitialization</b>: <b>cache is self-initializing</b>. Read-only. Reset: Fixed,1. 1=Cache is self initializing; cache does not need software initialization.</td></tr>
<tr><td rowspan="2">7:5</td><td><b>CacheLevel</b>: <b>cache level</b>. Read-only. Reset: Fixed,1h. Identifies the cache level.</td></tr>
<tr><td><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></td></tr>
<tr><td rowspan="2">4:0</td><td><b>CacheType</b>: <b>cache type</b>. Read-only. Reset: Fixed,01h. Identifies the type of cache.</td></tr>
<tr><td><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></td></tr>
</table>

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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EAX_x01

Bits	Description
31:26	Reserved.
25:14	**NumSharingCache**: **number of logical processors sharing cache**. Read-only. Reset: XXXh. See Core::X86::Cpuid::CachePropEax0[NumSharingCache].
13:10	Reserved.
9	**FullyAssociative**: **fully associative cache**. Read-only. Reset: Fixed,0. See Core::X86::Cpuid::CachePropEax0[FullyAssociative].
8	**SelfInitialization**: **cache is self-initializing**. Read-only. Reset: Fixed,1. See Core::X86::Cpuid::CachePropEax0[SelfInitialization].
7:5	**CacheLevel**: **cache level**. Read-only. Reset: Fixed,1h. Identifies the cache level. See Core::X86::Cpuid::CachePropEax0[CacheLevel].
4:0	**CacheType**: **cache type**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EAX_x02

Bits	Description
31:26	Reserved.
25:14	**NumSharingCache**: **number of logical processors sharing cache**. Read-only. Reset: XXXh. Core::X86::Cpuid::CachePropEax0[NumSharingCache].
13:10	Reserved.
9	**FullyAssociative**: **fully associative cache**. Read-only. Reset: Fixed,0.

	Core::X86::Cpuid::CachePropEax0[FullyAssociative].
8	**SelfInitialization**: **cache is self-initializing**. Read-only. Reset: Fixed,1. Core::X86::Cpuid::CachePropEax0[SelfInitialization].
7:5	**CacheLevel**: **cache level**. Read-only. Reset: Fixed,2h. Identifies the cache level. Core::X86::Cpuid::CachePropEax0[CacheLevel].
4:0	**CacheType**: **cache type**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EAX_x03

Bits	Description
31:26	Reserved.
25:14	**NumSharingCache**: **number of logical processors sharing cache**. Read-only. Reset: XXXh. The number of logical processors sharing this cache is NumSharingCache+1.
13:10	Reserved.
9	**FullyAssociative**: **fully associative cache**. Read-only. Reset: Fixed,0. Core::X86::Cpuid::CachePropEax0[FullyAssociative].
8	**SelfInitialization**: **cache is self-initializing**. Read-only. Reset: Fixed,1. Core::X86::Cpuid::CachePropEax0[SelfInitialization].
7:5	**CacheLevel**: **cache level**. Read-only. Reset: Fixed,3h. Identifies the cache level. Core::X86::Cpuid::CachePropEax0[CacheLevel].
4:0	**CacheType**: **cache type**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EAX_x04

Bits	Description
31:5	Reserved.
4:0	**CacheType**: **cache type**. 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,01C0_003Fh.

Core::X86::Cpuid::CachePropEbx0 reports topology information for the DC. See Core::X86::Cpuid::CachePropEax0.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EBX_x00

Bits	Description
31:22	**CacheNumWays**: **cache number of ways**. Read-only. Reset: Fixed,007h. Cache number of ways is CacheNumWays + 1.
21:12	**CachePhysPartitions**: **cache physical line partitions**. Read-only. Reset: Fixed,000h. Cache partitions is CachePhysPartitions + 1.
11:0	**CacheLineSize**: **cache line size in bytes**. 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,00C0_003Fh.

Core::X86::Cpuid::CachePropEbx1 reports topology information for the IC. See Core::X86::Cpuid::CachePropEax0.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EBX_x01

Bits	Description
31:22	**CacheNumWays**: **cache number of ways**. Read-only. Reset: Fixed,003h. Core::X86::Cpuid::CachePropEbx0[CacheNumWays].
21:12	**CachePhysPartitions**: **cache physical line partitions**. Read-only. Reset: Fixed,000h.

	Core::X86::Cpuid::CachePropEbx0[CachePhysPartitions].
11:0	**CacheLineSize**: **cache line size in bytes**. 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,01C0_003Fh.

Core::X86::Cpuid::CachePropEbx2 reports topology information for the L2. See Core::X86::Cpuid::CachePropEax0.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EBX_x02

Bits	Description
31:22	**CacheNumWays**: **cache number of ways**. Read-only. Reset: Fixed,007h. See Core::X86::Cpuid::CachePropEbx0[CacheNumWays].
21:12	**CachePhysPartitions**: **cache physical line partitions**. Read-only. Reset: Fixed,000h. See Core::X86::Cpuid::CachePropEbx0[CachePhysPartitions].
11:0	**CacheLineSize**: **cache line size in bytes**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EBX_x03

Bits	Description
31:22	**CacheNumWays**: **cache number of ways**. Read-only. Reset: Fixed,00Fh. See Core::X86::Cpuid::CachePropEbx0[CacheNumWays].
21:12	**CachePhysPartitions**: **cache physical line partitions**. Read-only. Reset: Fixed,000h. See Core::X86::Cpuid::CachePropEbx0[CachePhysPartitions].
11:0	**CacheLineSize**: **cache line size in bytes**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EBX_x04

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

CPUID_Fn8000001D_ECX_x00 [Cache Properties (DC)] (Core::X86::Cpuid::CachePropEcx0)

Read-only. Reset: Fixed,0000_003Fh.

Core::X86::Cpuid::CachePropEcx0 reports topology information for the DC. See Core::X86::Cpuid::CachePropEax0.

_lthree[1:0]_core[3:0]_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::CachePropEcx1)

Read-only. Reset: Fixed,0000_00FFh.

Core::X86::Cpuid::CachePropEcx1 reports topology information for the IC. See Core::X86::Cpuid::CachePropEax0.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_ECX_x01

Bits	Description
31:0	**CacheNumSets**: **cache number of sets**. Read-only. Reset: Fixed,0000_00FFh. See Core::X86::Cpuid::CachePropEcx0[CacheNumSets].

CPUID_Fn8000001D_ECX_x02 [Cache Properties (L2)] (Core::X86::Cpuid::CachePropEcx2)

Read-only. Reset: Fixed,0000_03FFh.

Core::X86::Cpuid::CachePropEcx2 reports topology information for the L2. See Core::X86::Cpuid::CachePropEax0.

_lthree[1:0]_core[3:0]_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::CachePropEcx0[CacheNumSets].

CPUID_Fn8000001D_ECX_x03 [Cache Properties (L3)] (Core::X86::Cpuid::CachePropEcx3)

Read-only.

Core::X86::Cpuid::CachePropEcx3 reports topology information for the L3.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_ECX_x03

Bits	Description
31:0	**CacheNumSets**: **cache number of sets**. Read-only. Reset: 0000_XXXXh. See Core::X86::Cpuid::CachePropEcx0[CacheNumSets].

ValidValues:

Value	Description
0000_0FFEh-0000_0000h	Reserved.
0000_0FFFh	4096 L3 Cache Sets.
0000_1FFEh-0000_1000h	Reserved.
0000_1FFFh	8192 L3 Cache Sets.
FFFF_FFFFh-0000_2000h	Reserved.

CPUID_Fn8000001D_ECX_x04 [Cache Properties Null] (Core::X86::Cpuid::CachePropEcx4)

Read-only. Reset: Fixed,0000_0000h.

Core::X86::Cpuid::CachePropEax3 reports done/null. See Core::X86::Cpuid::CachePropEax0.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_ECX_x04

Bits	Description
31:0	**CacheNumSets**. 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::CachePropEax0.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EDX_x00

Bits	Description
31:2	Reserved.
1	**CacheInclusive**: **cache inclusive**. Read-only. Reset: Fixed,0. 0=Cache is not inclusive of lower cache levels. 1=Cache is inclusive of lower cache levels.
0	**WBINVD**: **Write-Back Invalidate/Invalidate**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EDX_x01

Bits	Description
31:2	Reserved.
1	**CacheInclusive**: **cache inclusive**. Read-only. Reset: Fixed,0. See Core::X86::Cpuid::CachePropEdx0[CacheInclusive].
0	**WBINVD**: **Write-Back Invalidate/Invalidate**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EDX_x02

Bits	Description
31:2	Reserved.
1	**CacheInclusive**: **cache inclusive**. Read-only. Reset: Fixed,1. See Core::X86::Cpuid::CachePropEdx0[CacheInclusive].
0	**WBINVD**: **Write-Back Invalidate/Invalidate**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EDX_x03

Bits	Description
31:2	Reserved.
1	**CacheInclusive**: **cache inclusive**. Read-only. Reset: Fixed,0. See Core::X86::Cpuid::CachePropEdx0[CacheInclusive].
0	**WBINVD**: **Write-Back Invalidate/Invalidate**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001D_EDX_x04

Bits	Description
31:0	Reserved. Read-only. Reset: Fixed,0000_0000h.

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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001E_EAX

Bits	Description
31:0	**ExtendedApicId**: **extended APIC ID**. Read-only. See 2.1.12.2.1.3 [ApicId Enumeration Requirements].
	Reset: Core::X86::Msr::APIC_BAR[ApicEn] ? Fixed,{00_0000h , Core::X86::Apic::ApicId[ApicId]} :

	Fixed,0000_0000h.

CPUID_Fn8000001E_EBX [**Core Identifiers**] **(Core::X86::Cpuid::CoreId)**

Read-only.

See Core::X86::Cpuid::ExtApicId.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001E_EBX

Bits	Description
31:16	Reserved.
15:8	**ThreadsPerCore**: **threads per core**. Read-only. Reset: XXh. The number of threads per core is ThreadsPerCore+1.
7:0	**CoreId**: **core ID**. Read-only. Reset: Fixed,XXh.
	Description: For Family 17, Model 1, Revision 1 and later: CoreId = ({2'b0, DieId[1:0], LogicalComplexId[0], LogicalThreadId[2:0]} >> SMT).

CPUID_Fn8000001E_ECX [**Node Identifiers**] **(Core::X86::Cpuid::NodeId)**

Read-only.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001E_ECX

Bits	Description
31:11	Reserved.
10:8	**NodesPerProcessor**: **Node per processor**. Read-only. Reset: XXXb.
	ValidValues:

Value	Description
0h	1 node per processor.
1h	2 nodes per processor.
2h	Reserved.
3h	4 nodes per processor.
7h-4h	Reserved.

Bits	Description
7:0	**NodeId**: **Node ID**. Read-only. Reset: Fixed,XXh.
	Description: For Family 17, Model 1, Revision 1 and later: {5'b00000,1'b[SOCKET_ID],2'b[DIE_ID]}.

CPUID_Fn8000001F_EAX [**AMD Secure Encryption EAX**] **(Core::X86::Cpuid::SecureEncryptionEax)**

Read-only. Reset: Fixed,0000_000Fh.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001F_EAX

Bits	Description
31:4	Reserved.
3	**SevEs**. Read-only. Reset: Fixed,1. Secure Encrypted ES.
2	**VmPgFlush**: **VM Page Flush MSR is supported**. Read-only. Reset: Fixed,1. See Core::X86::Msr::VMPAGE_FLUSH.
1	**SEV**. Read-only. Reset: Fixed,1. Secure Encrypted Virtualization supported.
0	**SME**. Read-only. Reset: Fixed,1. Secure Memory Encryption supported.

CPUID_Fn8000001F_EBX [**AMD Secure Encryption EBX**] **(Core::X86::Cpuid::SecureEncryptionEbx)**

Read-only. Reset: 0000_016Fh.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001F_EBX

Bits	Description
31:12	Reserved.
11:6	**MemEncryptPhysAddWidth**. Read-only. Reset: 05h. Reduction of physical address space in bits when memory encryption is enabled (0 indicates no reduction).
	ValidValues:

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.
	3Fh-06h	Reserved.
5:0	**CBit**. Read-only. Reset: 2Fh. Page table bit number used to enable memory encryption.	

CPUID_Fn8000001F_ECX [AMD Secure Encryption ECX] (Core::X86::Cpuid::SecureEncryptionEcx)

Read-only. Reset: 0000_000Fh.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001F_ECX

Bits	Description
31:0	**NumEncryptedGuests**. Read-only. Reset: 0000_000Fh. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; CPUID_Fn8000001F_EDX

Bits	Description
31:0	**MinimumSEVASID**: **Minimum SEV enabled, SEV-ES disabled ASID**. Read-only. Reset: 0000_000Xh. Indicates the minimum ASID value that must be used for an SEV-enabled, SEV-ES-disabled guest.

2.1.14 MSR Registers

2.1.14.1 MSRs - MSR0000_xxxx

See 1.3.3 [Register Mnemonics] for a description of the register naming convention. MSRs are accessed through x86 WRMSR and RDMSR instructions.

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.

_lthree[1:0]_core[3:0]_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)

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_001B

Bits	Description
63:48	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0000h.
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 2.1.12.2.1.2 [APIC Register Space].
11	**ApicEn**: **APIC enable**. Read-write. Reset: 0. 0=Disable Local Apic. 1=Local APIC is enabled in xAPIC mode. See 2.1.12.2.1.2 [APIC Register Space].
10	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0.

9	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0.
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. Read-only,Error-on-write-1. Reset: Fixed,00h.

MSR0000_002A [Cluster ID] (Core::X86::Msr::EBL_CR_POWERON)

Writes to this register result in a GP fault with error code 0.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_002A

Bits	Description
63:18	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0000_0000_0000h.
17:16	**ClusterID**. Read,Error-on-write. Reset: 0h. The field does not affect hardware.
15:0	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0000h.

MSR0000_008B [Patch Level] (Core::X86::Msr::PATCH_LEVEL)

Read,Error-on-write,Volatile. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]; MSR0000_008B

Bits	Description
63:32	Reserved.
31:0	**PatchLevel**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_00E7

Bits	Description
63:0	**MPERF**: **maximum core clocks counter**. 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], 2.1.5 [Effective Frequency]

MSR0000_00E8 [Actual Performance Frequency Clock Count] (Core::X86::Msr::APERF)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_00E8

Bits	Description
63:0	**APERF**: **actual core clocks counter**. 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.

_lthree[1:0]_core[3:0]; MSR0000_00FE

Bits	Description
63:11	Reserved.
10	**MtrrCapWc**: **write-combining memory type**. Read,Error-on-write. Reset: 1. 1=The write combining memory type is supported.
9	Reserved.
8	**MtrrCapFix**: **fixed range register**. Read,Error-on-write. Reset: 1. 1=Fixed MTRRs are supported.
7:0	**MtrrCapVCnt**: **variable range registers count**. Read,Error-on-write. Reset: 08h. Specifies the number of variable MTRRs supported.

MSR0000_0174 [SYSENTER CS] (Core::X86::Msr::SYSENTER_CS)

Read-write. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_0174

Bits	Description
63:16	Reserved.
15:0	**SysEnterCS**: **SYSENTER target CS**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_0175

Bits	Description
63:32	Reserved.
31:0	**SysEnterESP**: **SYSENTER target SP**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_0176

Bits	Description
63:32	Reserved.
31:0	**SysEnterEIP**: **SYSENTER target IP**. Read-write. Reset: 0000_0000h. Holds the called procedure instruction pointer.

MSR0000_0179 [Global Machine Check Capabilities] (Core::X86::Msr::MCG_CAP)

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_0179

Bits	Description
63:9	Reserved.
8	**McgCtlP**: **MCG_CTL register present**. 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	**Count**. Read-only,Error-on-write,Volatile. Reset: XXh. Indicates the number of error reporting banks visible to each 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].

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_017A

Bits	Description
63:3	Reserved.
2	**MCIP**: **machine check in progress**. Read-write,Volatile. Reset: 0. 1=A machine check is in progress. Machine check progress.
1	**EIPV**: **error instruction pointer valid**. 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	**RIPV**: **restart instruction pointer valid**. 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)

Read-write. Reset: 0000_0000_0000_0000h.

This register controls enablement of the individual error reporting banks; see 3.1 [Machine Check Architecture]. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_017B

Bits	Description

63:23	Reserved.
22:0	**MCnEn**. Read-write. Reset: 00_0000h. 1=The MC0 machine check register bank is enabled.

MSR0000_01D9 [Debug Control] (Core::X86::Msr::DBG_CTL_MSR)

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_01D9

Bits	Description
63:7	Reserved.
6	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0.
5:2	**PB**: **performance monitor pin control**. Read-write. Reset: 0h. This field does not control any hardware.
1	**BTF**. Read-write. Reset: 0. 1=Enable branch single step.
0	**LBR**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_01DB

Bits	Description
63:0	**LastBranchFromIP**. Read,Error-on-write,Volatile. Reset: 0000_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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_01DC

Bits	Description
63:0	**LastBranchToIP**. Read,Error-on-write,Volatile. Reset: 0000_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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_01DD

Bits	Description
63:0	**LastIntFromIP**. Read,Error-on-write,Volatile. Reset: 0000_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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_01DE

Bits	Description
63:0	**LastIntToIP**. Read,Error-on-write,Volatile. Reset: 0000_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::Msr::MtrrVarBase)

Each MTRR (Core::X86::Msr::MtrrVarBase, Core::X86::Msr::MtrrFix_64K through Core::X86::Msr::MtrrFix_4K_7, or Core::X86::Msr::MTRRdefType) specifies a physical address range and a corresponding memory type (MemType) associated with that range. Setting the memory type to an unsupported value results 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::Msr::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:
CPUAddr[47:12] & PhyMask[47:12] == PhyBase[47:12] & 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.

_lthree[1:0]_core[3:0]_n0; MSR0000_0200
_lthree[1:0]_core[3:0]_n1; MSR0000_0202
_lthree[1:0]_core[3:0]_n2; MSR0000_0204
_lthree[1:0]_core[3:0]_n3; MSR0000_0206
_lthree[1:0]_core[3:0]_n4; MSR0000_0208
_lthree[1:0]_core[3:0]_n5; MSR0000_020A
_lthree[1:0]_core[3:0]_n6; MSR0000_020C
_lthree[1:0]_core[3:0]_n7; MSR0000_020E

Bits	Description
63:48	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0000h.
47:12	**PhyBase**: **base address**. Read-write. Reset: X_XXXX_XXXXh.
11:3	Reserved. Read-only,Error-on-write-1. Reset: Fixed,000h.
2:0	**MemType**: **memory type**. Read-write. Reset: XXXb. Address range from 00000h to 0FFFFh.
	ValidValues:

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::Msr::MtrrVarMask)

_lthree[1:0]_core[3:0]_n0; MSR0000_0201
_lthree[1:0]_core[3:0]_n1; MSR0000_0203
_lthree[1:0]_core[3:0]_n2; MSR0000_0205
_lthree[1:0]_core[3:0]_n3; MSR0000_0207
_lthree[1:0]_core[3:0]_n4; MSR0000_0209
_lthree[1:0]_core[3:0]_n5; MSR0000_020B
_lthree[1:0]_core[3:0]_n6; MSR0000_020D
_lthree[1:0]_core[3:0]_n7; MSR0000_020F

Bits	Description
63:48	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0000h.
47:12	**PhyMask**: **address mask**. Read-write. Reset: X_XXXX_XXXXh.
11	**Valid**: **valid**. Read-write. Reset: X. 1=The variable-size MTRR pair is enabled.
10:0	Reserved. Read-only,Error-on-write-1. Reset: Fixed,000h.

MSR0000_0250 [Fixed-Size MTRRs] (Core::X86::Msr::MtrrFix_64K)

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.

_lthree[1:0]_core[3:0]_nSIZE64K; MSR0000_0250

Bits	Description
63:61	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
60	**RdDram_64K_70000**: **read DRAM**. 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	**WrDram_64K_70000**: **write DRAM**. 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	**MemType_64K_70000**: **memory type**. Read-write. Reset: XXXb.	
	ValidValues:	
	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. Read-only,Error-on-write-1. Reset: Fixed,0h.	
52	**RdDram_64K_60000**: **read DRAM**. 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	**WrDram_64K_60000**: **write DRAM**. 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	**MemType_64K_60000**: **memory type**. Read-write. Reset: XXXb.	
	ValidValues:	
	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. Read-only,Error-on-write-1. Reset: Fixed,0h.	
44	**RdDram_64K_50000**: **read DRAM**. 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	**WrDram_64K_50000**: **write DRAM**. 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	**MemType_64K_50000**: **memory type**. Read-write. Reset: XXXb.	
	ValidValues:	
	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.

<table>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">36</td><td><b>RdDram_64K_40000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">35</td><td><b>WrDram_64K_40000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">34:32</td><td><b>MemType_64K_40000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">28</td><td><b>RdDram_64K_30000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">27</td><td><b>WrDram_64K_30000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">26:24</td><td><b>MemType_64K_30000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>23:21</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">20</td><td><b>RdDram_64K_20000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">19</td><td><b>WrDram_64K_20000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">18:16</td><td><b>MemType_64K_20000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><b>ValidValues</b>:</td></tr>
</table>

<table>
<tr><td></td><td><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></td></tr>
<tr><td>15:13</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">12</td><td><b>RdDram_64K_10000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">11</td><td><b>WrDram_64K_10000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">10:8</td><td><b>MemType_64K_10000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>7:5</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">4</td><td><b>RdDram_64K_00000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">3</td><td><b>WrDram_64K_00000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">2:0</td><td><b>MemType_64K_00000</b>: <b>memory type</b>. Read-write. Reset: XXXb. Address range from 00000h to 0FFFFh.</td></tr>
<tr><td><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></td></tr>
</table>

MSR0000_0258 [Fixed-Size MTRRs] (Core::X86::Msr::MtrrFix_16K_0)

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.

_lthree[1:0]_core[3:0]_nSIZE16K0; MSR0000_0258

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:61</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>60</td><td><b>RdDram_16K_9C000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>59</td><td><b>WrDram_16K_9C000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>58:56</td><td><b>MemType_16K_9C000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>55:53</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>52</td><td><b>RdDram_16K_98000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>51</td><td><b>WrDram_16K_98000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>50:48</td><td><b>MemType_16K_98000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>47:45</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>44</td><td><b>RdDram_16K_94000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
</table>

<table>
<tr><td>43</td><td><b>WrDram_16K_94000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>42:40</td><td><b>MemType_16K_94000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>36</td><td><b>RdDram_16K_90000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>35</td><td><b>WrDram_16K_90000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>34:32</td><td><b>MemType_16K_90000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>28</td><td><b>RdDram_16K_8C000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>27</td><td><b>WrDram_16K_8C000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>26:24</td><td><b>MemType_16K_8C000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<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></td></tr>
<tr><td>23:21</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">20</td><td><b>RdDram_16K_88000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">19</td><td><b>WrDram_16K_88000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">18:16</td><td><b>MemType_16K_88000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>15:13</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">12</td><td><b>RdDram_16K_84000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">11</td><td><b>WrDram_16K_84000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">10:8</td><td><b>MemType_16K_84000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>7:5</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">4</td><td><b>RdDram_16K_80000</b>: <b>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::Msr::SYS_CFG[MtrrFixDramEn,MtrrFixDramModEn] masks reads of the stored value.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>3</td><td><b>WrDram_16K_80000</b>: <b>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.</td></tr>
</table>

	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	**MemType_16K_80000**: **memory type**. Read-write. Reset: XXXb. Address range from 80000h to 83FFFh.
	ValidValues:

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.

_lthree[1:0]_core[3:0]_nSIZE16K1; MSR0000_0259

Bits	Description
63:61	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
60	**RdDram_16K_BC000**: **read DRAM**. 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	**WrDram_16K_BC000**: **write DRAM**. 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	**MemType_16K_BC000**: **memory type**. Read-write. Reset: XXXb.
	ValidValues:

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.

Bits	Description
55:53	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
52	**RdDram_16K_B8000**: **read DRAM**. 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	**WrDram_16K_B8000**: **write DRAM**. 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	**MemType_16K_B8000**: **memory type**. Read-write. Reset: XXXb.

<table>
<tr><td></td><td><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></td></tr>
<tr><td>47:45</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>44</td><td><b>RdDram_16K_B4000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>43</td><td><b>WrDram_16K_B4000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>42:40</td><td><b>MemType_16K_B4000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>36</td><td><b>RdDram_16K_B0000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>35</td><td><b>WrDram_16K_B0000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>34:32</td><td><b>MemType_16K_B0000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>28</td><td><b>RdDram_16K_AC000</b>: <b>read DRAM</b>. 0=Read accesses to the range are marked as MMIO. 1=Read accesses to</td></tr>
</table>

<table>
<tr><td></td><td>the range are marked as destined for DRAM.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>27</td><td><b>WrDram_16K_AC000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>26:24</td><td><b>MemType_16K_AC000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>23:21</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>20</td><td><b>RdDram_16K_A8000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>19</td><td><b>WrDram_16K_A8000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>18:16</td><td><b>MemType_16K_A8000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>15:13</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>12</td><td><b>RdDram_16K_A4000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>11</td><td><b>WrDram_16K_A4000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>10:8</td><td><b>MemType_16K_A4000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>UC or uncacheable.</td></tr>
</table></td></tr>
</table>

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. Read-only,Error-on-write-1. Reset: Fixed,0h.
4	**RdDram_16K_A0000**: **read DRAM**. 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	**WrDram_16K_A0000**: **write DRAM**. 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	**MemType_16K_A0000**: **memory type**. Read-write. Reset: XXXb. Address range from A0000h to A3FFFh.
	ValidValues:

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::Msr::MtrrFix_4K_0)**

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.

_lthree[1:0]_core[3:0]_nSIZE4K0; MSR0000_0268

Bits	Description
63:61	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
60	**RdDram_4K_C7000**: **read DRAM**. 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	**WrDram_4K_C7000**: **write DRAM**. 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	**MemType_4K_C7000**: **memory type**. Read-write. Reset: XXXb.
	ValidValues:

Value	Description
0h	UC or uncacheable.
1h	WC or write combining.
3h-2h	Reserved.

<table>
<tr><td rowspan="4"></td><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>
<tr><td>55:53</td><td colspan="2">Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">52</td><td colspan="2"><b>RdDram_4K_C6000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">51</td><td colspan="2"><b>WrDram_4K_C6000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="10">50:48</td><td colspan="2"><b>MemType_4K_C6000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td colspan="2"><b>ValidValues</b>:</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></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>
<tr><td>47:45</td><td colspan="2">Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">44</td><td colspan="2"><b>RdDram_4K_C5000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">43</td><td colspan="2"><b>WrDram_4K_C5000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="10">42:40</td><td colspan="2"><b>MemType_4K_C5000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td colspan="2"><b>ValidValues</b>:</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></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>
<tr><td>39:37</td><td colspan="2">Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">36</td><td colspan="2"><b>RdDram_4K_C4000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>35</td><td colspan="2"><b>WrDram_4K_C4000</b>: <b>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.</td></tr>
</table>

<table>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>34:32</td><td><b>MemType_4K_C4000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>28</td><td><b>RdDram_4K_C3000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>27</td><td><b>WrDram_4K_C3000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>26:24</td><td><b>MemType_4K_C3000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>23:21</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>20</td><td><b>RdDram_4K_C2000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>19</td><td><b>WrDram_4K_C2000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>18:16</td><td><b>MemType_4K_C2000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td>7h</td><td>Reserved.</td></tr></table></td></tr>
<tr><td>15:13</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">12</td><td><b>RdDram_4K_C1000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">11</td><td><b>WrDram_4K_C1000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">10:8</td><td><b>MemType_4K_C1000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>7:5</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">4</td><td><b>RdDram_4K_C0000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">3</td><td><b>WrDram_4K_C0000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">2:0</td><td><b>MemType_4K_C0000</b>: <b>memory type</b>. Read-write. Reset: XXXb. Address range from C0000h to C0FFFh.</td></tr>
<tr><td><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></td></tr>
</table>

MSR0000_0269 [Fixed-Size MTRRs] (Core::X86::Msr::MtrrFix_4K_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.

_lthree[1:0]_core[3:0]_nSIZE4K1; MSR0000_0269

Bits	Description
63:61	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.

<table>
<tr><td rowspan="2">60</td><td><b>RdDram_4K_CF000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">59</td><td><b>WrDram_4K_CF000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">58:56</td><td><b>MemType_4K_CF000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>55:53</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">52</td><td><b>RdDram_4K_CE000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">51</td><td><b>WrDram_4K_CE000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">50:48</td><td><b>MemType_4K_CE000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>47:45</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">44</td><td><b>RdDram_4K_CD000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">43</td><td><b>WrDram_4K_CD000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">42:40</td><td><b>MemType_4K_CD000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<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></td></tr>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>36</td><td><b>RdDram_4K_CC000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>35</td><td><b>WrDram_4K_CC000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>34:32</td><td><b>MemType_4K_CC000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>28</td><td><b>RdDram_4K_CB000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>27</td><td><b>WrDram_4K_CB000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>26:24</td><td><b>MemType_4K_CB000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>23:21</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>20</td><td><b>RdDram_4K_CA000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset:</td></tr>
</table>

<table>
<tr><td></td><td>Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>19</td><td><b>WrDram_4K_CA000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>18:16</td><td><b>MemType_4K_CA000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>15:13</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>12</td><td><b>RdDram_4K_C9000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>11</td><td><b>WrDram_4K_C9000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>10:8</td><td><b>MemType_4K_C9000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>7:5</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>4</td><td><b>RdDram_4K_C8000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>3</td><td><b>WrDram_4K_C8000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>2:0</td><td><b>MemType_4K_C8000</b>: <b>memory type</b>. Read-write. Reset: XXXb. Address range from C8000 to C8FFF.</td></tr>
<tr><td></td><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>UC or uncacheable.</td></tr>
</table></td></tr>
</table>

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::Msr::MtrrFix_4K_2)

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.

_lthree[1:0]_core[3:0]_nSIZE4K2; MSR0000_026A

Bits	Description
63:61	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
60	**RdDram_4K_D7000**: **read DRAM**. 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	**WrDram_4K_D7000**: **write DRAM**. 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	**MemType_4K_D7000**: **memory type**. Read-write. Reset: XXXb.

ValidValues:

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.

Bits	Description
55:53	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
52	**RdDram_4K_D6000**: **read DRAM**. 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	**WrDram_4K_D6000**: **write DRAM**. 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	**MemType_4K_D6000**: **memory type**. Read-write. Reset: XXXb.

ValidValues:

Value	Description
0h	UC or uncacheable.
1h	WC or write combining.
3h-2h	Reserved.
4h	WT or write through.
5h	WP or write protect.

<table>
<tr><td></td><td><table><tr><td>6h</td><td>WB or write back.</td></tr><tr><td>7h</td><td>Reserved.</td></tr></table></td></tr>
<tr><td>47:45</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>44</td><td><b>RdDram_4K_D5000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>43</td><td><b>WrDram_4K_D5000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>42:40</td><td><b>MemType_4K_D5000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>36</td><td><b>RdDram_4K_D4000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>35</td><td><b>WrDram_4K_D4000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>34:32</td><td><b>MemType_4K_D4000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>28</td><td><b>RdDram_4K_D3000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>27</td><td><b>WrDram_4K_D3000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
</table>

26:24	**MemType_4K_D3000**: **memory type**. Read-write. Reset: XXXb.	
	ValidValues:	
	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.
23:21	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.	
20	**RdDram_4K_D2000**: **read DRAM**. 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	**WrDram_4K_D2000**: **write DRAM**. 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	**MemType_4K_D2000**: **memory type**. Read-write. Reset: XXXb.	
	ValidValues:	
	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.
15:13	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.	
12	**RdDram_4K_D1000**: **read DRAM**. 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	**WrDram_4K_D1000**: **write DRAM**. 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	**MemType_4K_D1000**: **memory type**. Read-write. Reset: XXXb.	
	ValidValues:	
	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.
7:5	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.	

<table>
<tr><td>4</td><td><b>RdDram_4K_D0000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>3</td><td><b>WrDram_4K_D0000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>2:0</td><td><b>MemType_4K_D0000</b>: <b>memory type</b>. Read-write. Reset: XXXb. Address range from D0000h to D0FFFh.</td></tr>
<tr><td></td><td><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></td></tr>
</table>

MSR0000_026B [Fixed-Size MTRRs] (Core::X86::Msr::MtrrFix_4K_3)

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.

_lthree[1:0]_core[3:0]_nSIZE4K3; MSR0000_026B

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:61</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>60</td><td><b>RdDram_4K_DF000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>59</td><td><b>WrDram_4K_DF000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>58:56</td><td><b>MemType_4K_DF000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>55:53</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>52</td><td><b>RdDram_4K_DE000</b>: <b>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.</td></tr>
</table>

<table>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>51</td><td><b>WrDram_4K_DE000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>50:48</td><td><b>MemType_4K_DE000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>47:45</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>44</td><td><b>RdDram_4K_DD000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>43</td><td><b>WrDram_4K_DD000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>42:40</td><td><b>MemType_4K_DD000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>36</td><td><b>RdDram_4K_DC000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>35</td><td><b>WrDram_4K_DC000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>34:32</td><td><b>MemType_4K_DC000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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>
</table></td></tr>
</table>

	3h-2h	Reserved.
	4h	WT or write through.
	5h	WP or write protect.
	6h	WB or write back.
	7h	Reserved.
31:29	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.	
28	**RdDram_4K_DB000**: **read DRAM**. 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	**WrDram_4K_DB000**: **write DRAM**. 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	**MemType_4K_DB000**: **memory type**. Read-write. Reset: XXXb.	
	ValidValues:	
	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.
23:21	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.	
20	**RdDram_4K_DA000**: **read DRAM**. 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	**WrDram_4K_DA000**: **write DRAM**. 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	**MemType_4K_DA000**: **memory type**. Read-write. Reset: XXXb.	
	ValidValues:	
	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.
15:13	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.	
12	**RdDram_4K_D9000**: **read DRAM**. 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	**WrDram_4K_D9000**: **write DRAM**. 0=Write accesses to the range are marked as MMIO. 1=Write accesses to	

<table>
<tr><td></td><td>the range are marked as destined for DRAM.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>10:8</td><td><b>MemType_4K_D9000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>7:5</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>4</td><td><b>RdDram_4K_D8000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>3</td><td><b>WrDram_4K_D8000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>2:0</td><td><b>MemType_4K_D8000</b>: <b>memory type</b>. Read-write. Reset: XXXb. Address range from D8000h to D8FFFh.</td></tr>
<tr><td></td><td><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></td></tr>
</table>

MSR0000_026C [Fixed-Size MTRRs] (Core::X86::Msr::MtrrFix_4K_4)

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.

_lthree[1:0]_core[3:0]_nSIZE4K4; MSR0000_026C

Bits	Description
63:61	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
60	**RdDram_4K_E7000**: **read DRAM**. 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	**WrDram_4K_E7000**: **write DRAM**. 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:

<table>
<tr><td></td><td>Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>58:56</td><td><b>MemType_4K_E7000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>55:53</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>52</td><td><b>RdDram_4K_E6000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>51</td><td><b>WrDram_4K_E6000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>50:48</td><td><b>MemType_4K_E6000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>47:45</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>44</td><td><b>RdDram_4K_E5000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>43</td><td><b>WrDram_4K_E5000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>42:40</td><td><b>MemType_4K_E5000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
</table>

<table>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">36</td><td><b>RdDram_4K_E4000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">35</td><td><b>WrDram_4K_E4000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">34:32</td><td><b>MemType_4K_E4000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">28</td><td><b>RdDram_4K_E3000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">27</td><td><b>WrDram_4K_E3000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">26:24</td><td><b>MemType_4K_E3000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>23:21</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">20</td><td><b>RdDram_4K_E2000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">19</td><td><b>WrDram_4K_E2000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">18:16</td><td><b>MemType_4K_E2000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><b>ValidValues</b>:</td></tr>
</table>

<table>
<tr><td></td><td><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></td></tr>
<tr><td>15:13</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>12</td><td><b>RdDram_4K_E1000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>11</td><td><b>WrDram_4K_E1000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>10:8</td><td><b>MemType_4K_E1000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>7:5</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>4</td><td><b>RdDram_4K_E0000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>3</td><td><b>WrDram_4K_E0000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>2:0</td><td><b>MemType_4K_E0000</b>: <b>memory type</b>. Read-write. Reset: XXXb. Address range from E0000h to E0FFFh.</td></tr>
<tr><td></td><td><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></td></tr>
</table>

MSR0000_026D [Fixed-Size MTRRs] (Core::X86::Msr::MtrrFix_4K_5)

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.

_lthree[1:0]_core[3:0]_nSIZE4K5; MSR0000_026D

Bits	Description
63:61	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
60	**RdDram_4K_EF000**: **read DRAM**. 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	**WrDram_4K_EF000**: **write DRAM**. 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	**MemType_4K_EF000**: **memory type**. Read-write. Reset: XXXb.
	ValidValues:

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.

Bits	Description
55:53	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
52	**RdDram_4K_EE000**: **read DRAM**. 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	**WrDram_4K_EE000**: **write DRAM**. 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	**MemType_4K_EE000**: **memory type**. Read-write. Reset: XXXb.
	ValidValues:

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.

Bits	Description
47:45	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
44	**RdDram_4K_ED000**: **read DRAM**. 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.

<table>
<tr><td rowspan="2">43</td><td><b>WrDram_4K_ED000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">42:40</td><td><b>MemType_4K_ED000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">36</td><td><b>RdDram_4K_EC000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">35</td><td><b>WrDram_4K_EC000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">34:32</td><td><b>MemType_4K_EC000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">28</td><td><b>RdDram_4K_EB000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">27</td><td><b>WrDram_4K_EB000</b>: <b>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.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">26:24</td><td><b>MemType_4K_EB000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td><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>
</table></td></tr>
</table>

<table>
<tr><td></td><td>5h</td><td>WP or write protect.</td></tr>
<tr><td></td><td>6h</td><td>WB or write back.</td></tr>
<tr><td></td><td>7h</td><td>Reserved.</td></tr>
<tr><td>23:21</td><td colspan="2">Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">20</td><td colspan="2"><b>RdDram_4K_EA000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">19</td><td colspan="2"><b>WrDram_4K_EA000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="10">18:16</td><td colspan="2"><b>MemType_4K_EA000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td colspan="2"><b>ValidValues</b>:</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></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>
<tr><td>15:13</td><td colspan="2">Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">12</td><td colspan="2"><b>RdDram_4K_E9000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="2">11</td><td colspan="2"><b>WrDram_4K_E9000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td rowspan="10">10:8</td><td colspan="2"><b>MemType_4K_E9000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td colspan="2"><b>ValidValues</b>:</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></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>
<tr><td>7:5</td><td colspan="2">Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td rowspan="2">4</td><td colspan="2"><b>RdDram_4K_E8000</b>: <b>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.</td></tr>
<tr><td colspan="2">AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>3</td><td colspan="2"><b>WrDram_4K_E8000</b>: <b>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.</td></tr>
</table>

	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	**MemType_4K_E8000**: **memory type**. Read-write. Reset: XXXb. Address range from E8000h to E8FFFh.
	ValidValues:

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::Msr::MtrrFix_4K_6)

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.

_lthree[1:0]_core[3:0]_nSIZE4K6; MSR0000_026E

Bits	Description
63:61	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
60	**RdDram_4K_F7000**: **read DRAM**. 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	**WrDram_4K_F7000**: **write DRAM**. 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	**MemType_4K_F7000**: **memory type**. Read-write. Reset: XXXb.
	ValidValues:

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.

Bits	Description
55:53	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
52	**RdDram_4K_F6000**: **read DRAM**. 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	**WrDram_4K_F6000**: **write DRAM**. 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	**MemType_4K_F6000**: **memory type**. Read-write. Reset: XXXb.

<table>
<tr><td></td><td><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></td></tr>
<tr><td>47:45</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>44</td><td><b>RdDram_4K_F5000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>43</td><td><b>WrDram_4K_F5000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>42:40</td><td><b>MemType_4K_F5000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>36</td><td><b>RdDram_4K_F4000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>35</td><td><b>WrDram_4K_F4000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>34:32</td><td><b>MemType_4K_F4000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>28</td><td><b>RdDram_4K_F3000</b>: <b>read DRAM</b>. 0=Read accesses to the range are marked as MMIO. 1=Read accesses to the</td></tr>
</table>

<table>
<tr><td></td><td>range are marked as destined for DRAM.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>27</td><td><b>WrDram_4K_F3000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>26:24</td><td><b>MemType_4K_F3000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>23:21</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>20</td><td><b>RdDram_4K_F2000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>19</td><td><b>WrDram_4K_F2000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>18:16</td><td><b>MemType_4K_F2000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>15:13</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>12</td><td><b>RdDram_4K_F1000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>11</td><td><b>WrDram_4K_F1000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>10:8</td><td><b>MemType_4K_F1000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>UC or uncacheable.</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<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></td></tr>
<tr><td>7:5</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>4</td><td><b>RdDram_4K_F0000</b>: <b>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.</td></tr>
<tr><td>3</td><td><b>WrDram_4K_F0000</b>: <b>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.</td></tr>
<tr><td>2:0</td><td><b>MemType_4K_F0000</b>: <b>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></td></tr>
</table>

MSR0000_026F [Fixed-Size MTRRs] (Core::X86::Msr::MtrrFix_4K_7)

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.

_lthree[1:0]_core[3:0]_nSIZE4K7; MSR0000_026F

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:61</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>60</td><td><b>RdDram_4K_FF000</b>: <b>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.</td></tr>
<tr><td>59</td><td><b>WrDram_4K_FF000</b>: <b>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.</td></tr>
<tr><td>58:56</td><td><b>MemType_4K_FF000</b>: <b>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>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<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></td></tr>
<tr><td>55:53</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>52</td><td><b>RdDram_4K_FE000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>51</td><td><b>WrDram_4K_FE000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>50:48</td><td><b>MemType_4K_FE000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>47:45</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>44</td><td><b>RdDram_4K_FD000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>43</td><td><b>WrDram_4K_FD000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>42:40</td><td><b>MemType_4K_FD000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>39:37</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>36</td><td><b>RdDram_4K_FC000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>35</td><td><b>WrDram_4K_FC000</b>: <b>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.</td></tr>
</table>

<table>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>34:32</td><td><b>MemType_4K_FC000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>31:29</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>28</td><td><b>RdDram_4K_FB000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>27</td><td><b>WrDram_4K_FB000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>26:24</td><td><b>MemType_4K_FB000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>23:21</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>20</td><td><b>RdDram_4K_FA000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>19</td><td><b>WrDram_4K_FA000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>18:16</td><td><b>MemType_4K_FA000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td>7h</td><td>Reserved.</td></tr></table></td></tr>
<tr><td>15:13</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>12</td><td><b>RdDram_4K_F9000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>11</td><td><b>WrDram_4K_F9000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>10:8</td><td><b>MemType_4K_F9000</b>: <b>memory type</b>. Read-write. Reset: XXXb.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>7:5</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.</td></tr>
<tr><td>4</td><td><b>RdDram_4K_F8000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>3</td><td><b>WrDram_4K_F8000</b>: <b>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.</td></tr>
<tr><td></td><td>AccessType: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? Read-write : Read,Error-on-write-1. Reset: Core::X86::Msr::SYS_CFG[MtrrFixDramModEn] ? X : Fixed,0.</td></tr>
<tr><td>2:0</td><td><b>MemType_4K_F8000</b>: <b>memory type</b>. Read-write. Reset: XXXb. Address range from F8000h to F8FFFh.</td></tr>
<tr><td></td><td><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></td></tr>
</table>

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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSR0000_0277

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:59</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,00h.</td></tr>
<tr><td>58:56</td><td><b>PA7MemType</b>. Read-write. Reset: 0h. Default UC. MemType for {PAT, PCD, PWT} = 7h.</td></tr>
<tr><td></td><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
</table></td></tr>
</table>

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. Read-only,Error-on-write-1. Reset: Fixed,00h.

50:48 **PA6MemType**. Read-write. Reset: 7h. Default UC. MemType for {PAT, PCD, PWT} = 6h.

ValidValues:

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.

47:43 Reserved. Read-only,Error-on-write-1. Reset: Fixed,00h.

42:40 **PA5MemType**. Read-write. Reset: 4h. Default WT. MemType for {PAT, PCD, PWT} = 5h.

ValidValues:

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. Read-only,Error-on-write-1. Reset: Fixed,00h.

34:32 **PA4MemType**. Read-write. Reset: 6h. Default WB. MemType for {PAT, PCD, PWT} = 4h.

ValidValues:

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. Read-only,Error-on-write-1. Reset: Fixed,00h.

26:24 **PA3MemType**. Read-write. Reset: 0h. Default UC. MemType for {PAT, PCD, PWT} = 3h.

ValidValues:

Value	Description
0h	UC or uncacheable.
1h	WC or write combining.
3h-2h	Reserved.
4h	WT or write through.

<table>
<tr><td></td><td><table>
<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></td></tr>
<tr><td>23:19</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,00h.</td></tr>
<tr><td>18:16</td><td><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>Reserved.</td></tr>
</table></td></tr>
<tr><td>15:11</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,00h.</td></tr>
<tr><td>10:8</td><td><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></td></tr>
<tr><td>7:3</td><td>Reserved. Read-only,Error-on-write-1. Reset: Fixed,00h.</td></tr>
<tr><td>2:0</td><td><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></td></tr>
</table>

MSR0000_02FF [MTRR Default Memory Type] (Core::X86::Msr::MTRRdefType)

See Core::X86::Msr::MtrrVarBase for general MTRR information.

_lthree[1:0]_core[3:0]; MSR0000_02FF

Bits	Description
63:12	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0_0000_0000_0000h.
11	**MtrrDefTypeEn**: **variable and fixed MTRR enable**. Read-write. Reset: 0. 0=Fixed and variable MTRRs are not enabled. 1=Core::X86::Msr::MtrrVarBase, and Core::X86::Msr::MtrrFix_64K through Core::X86::Msr::MtrrFix_4K_7 are enabled.
10	**MtrrDefTypeFixEn**: **fixed MTRR enable**. Read-write. Reset: 0. 0=Core::X86::Msr::MtrrFix_64K through Core::X86::Msr::MtrrFix_4K_7 are not enabled. 1=Core::X86::Msr::MtrrFix_64K through Core::X86::Msr::MtrrFix_4K_7 are enabled. This field is ignored (and the fixed MTRRs are not enabled) if Core::X86::Msr::MTRRdefType[MtrrDefTypeEn] == 0.

9:8	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0h.
7:0	**MemType**: **memory type**. Read-write. Reset: 00h.
	Description: 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::Msr::MtrrFix_64K through Core::X86::Msr::MtrrFix_4K_7[2:0]}. Other write values cause a GP(0).

2.1.14.2 MSRs - MSRC000_0xxx

See 1.3.3 [Register Mnemonics] for a description of the register naming convention. MSRs are accessed through x86 WRMSR and RDMSR instructions.

MSRC000_0080 [**Extended Feature Enable**] **(Core::X86::Msr::EFER)**

SKINIT Execution: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0080

Bits	Description
63:16	Reserved. Read-only,Error-on-write-1. Reset: Fixed,0000_0000_0000h.
15	**TCE**: **translation cache extension enable**. 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	**FFXSE**: **fast FXSAVE/FRSTOR enable**. 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	**LMSLE**: **long mode segment limit enable**. Read-write. Reset: 0. 1=Enables the long mode segment limit check mechanism.
12	**SVME**: **secure virtual machine (SVM) enable**. Reset: Fixed,0. 1=SVM features are enabled.
	AccessType: Core::X86::Msr::VM_CR[SvmeDisable] ? Read-only,Error-on-write-1 : Read-write.
11	**NXE**: **no-execute page enable**. Read-write. Reset: 0. 1=The no-execute page protection feature is enabled.
10	**LMA**: **long mode active**. 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. Read-only,Error-on-write-1. Reset: Fixed,0.
8	**LME**: **long mode enable**. Read-write. Reset: 0. 1=Long mode is enabled.
7:1	Reserved. Read-only. Reset: Fixed,00h.
0	**SYSCALL**: **system call extension enable**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0081

Bits	Description
63:48	**SysRetSel**: **SYSRET CS and SS**. Read-write. Reset: 0000h.
47:32	**SysCallSel**: **SYSCALL CS and SS**. Read-write. Reset: 0000h.
31:0	**Target**: **SYSCALL target address**. Read-write. Reset: 0000_0000h.

MSRC000_0082 [Long Mode SYSCALL Target Address] (Core::X86::Msr::STAR64)

Read-write. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0082

Bits	Description
63:0	**LSTAR**: **long mode target address**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0083

Bits	Description
63:0	**CSTAR**: **compatibility mode target address**. 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)

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0084

Bits	Description
63:32	Reserved. Read-only. Reset: Fixed,0000_0000h.
31:0	**Mask**: **SYSCALL flag mask**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_00E7

Bits	Description
63:0	**MPerfReadOnly**: **read-only maximum core clocks counter**. 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], 2.1.5 [Effective Frequency]. This register is not affected by writes to Core::X86::Msr::MPERF.
	AccessType: Core::X86::Msr::HWCR[EffFreqReadOnlyLock] ? Read-only,Volatile : Read-write,Volatile.

MSRC000_00E8 [Read-Only Actual Performance Frequency Clock Count] (Core::X86::Msr::APerfReadOnly)

Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_00E8

Bits	Description
63:0	**APerfReadOnly**: **read-only actual core clocks counter**. 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-only,Volatile : Read-write,Volatile.

MSRC000_00E9 [Instructions Retired Performance Count] (Core::X86::Msr::IRPerfCount)

Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_00E9

Bits	Description
63:0	**IRPerfCount**: **instructions retired counter**. Reset: 0000_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-only,Volatile : Read-write,Volatile.

MSRC000_0100 [FS Base] (Core::X86::Msr::FS_BASE)

Read-write. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0100

Bits	Description
63:0	**FSBase**: **expanded FS segment base**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0101

Bits	Description
63:0	**GSBase**: **expanded GS segment base**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0102

Bits	Description
63:0	**KernelGSBase**: **kernel data structure pointer**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0103

Bits	Description
63:32	Reserved.
31:0	**TscAux**: **auxiliary time stamp counter data**. 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 in a guest frequency matches the core P0 frequency.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC000_0104

Bits	Description
63:40	Reserved. Read-only,Error-on-write-1. Reset: Fixed,00_0000h.
39:32	**TscRateMsrInt**: **time stamp counter rate integer**. Read-write. Reset: 01h. Specifies the integer part of the MSR TSC ratio value.
31:0	**TscRateMsrFrac**: **time stamp counter rate fraction**. Read-write. Reset: 0000_0000h. Specifies the fractional part of the MSR TSC ratio value.

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	**ThresholdLvtOffset**. 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	**DeferredLvtOffset**. Read-write. Reset: 0h.
	Description: 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.

2.1.14.2.1 MSRs - MSRC000_2xxx

The MCA registers including the legacy aliases (MSR0000_000[1:0], MSR0000_04xx) are mapped to MSRC000_2xxx. See 3.2.5 [MCA Banks].

2.1.14.3 MSRs - MSRC001_0xxx

See 1.3.3 [Register Mnemonics] for a description of the register naming convention. MSRs are accessed through x86 WRMSR and RDMSR instructions.

MSRC001_000[0...3] [**Performance Event Select [3:0]**] **(Core::X86::Msr::PERF_LEGACY_CTL)**	
Read-write. Reset: 0000_0000_0000_0000h.	
The legacy alias of Core::X86::Msr::PERF_CTL. See Core::X86::Msr::PERF_CTL.	
_lthree[1:0]_core[3:0]_thread[1:0]_n0; MSRC001_0000	
_lthree[1:0]_core[3:0]_thread[1:0]_n1; MSRC001_0001	
_lthree[1:0]_core[3:0]_thread[1:0]_n2; MSRC001_0002	
_lthree[1:0]_core[3:0]_thread[1:0]_n3; MSRC001_0003	
Bits	**Description**
63:42	Reserved.
41:40	**HostGuestOnly**: **count only host/guest events**. Read-write. Reset: 0h.
39:36	Reserved.
35:32	**EventSelect[11:8]**: **performance event select**. Read-write. Reset: 0h.
31:24	**CntMask**: **counter mask**. Read-write. Reset: 00h.
23	**Inv**: **invert counter mask**. Read-write. Reset: 0.
22	**En**: **enable performance counter**. Read-write. Reset: 0.
21	Reserved.
20	**Int**: **enable APIC interrupt**. Read-write. Reset: 0.
19	Reserved.
18	**Edge**: **edge detect**. Read-write. Reset: 0.
17:16	**OsUserMode**: **OS and user mode**. Read-write. Reset: 0h.
15:8	**UnitMask**: **event qualification**. Read-write. Reset: 00h. When selecting an event for which not all UnitMask bits are defined, the undefined UnitMask bits should be set to zero.
7:0	**EventSelect[7:0]**: **event select**. Read-write. Reset: 00h.

MSRC001_000[4...7] [**Performance Event Counter [3:0]**] **(Core::X86::Msr::PERF_LEGACY_CTR)**

Read-write,Volatile. Reset: 0000_0000_0000_0000h.	
The legacy alias of Core::X86::Msr::PERF_CTR. See Core::X86::Msr::PERF_CTR.	
_lthree[1:0]_core[3:0]_thread[1:0]_n0; MSRC001_0004	
_lthree[1:0]_core[3:0]_thread[1:0]_n1; MSRC001_0005	
_lthree[1:0]_core[3:0]_thread[1:0]_n2; MSRC001_0006	
_lthree[1:0]_core[3:0]_thread[1:0]_n3; MSRC001_0007	
Bits	**Description**
63:48	Reserved.
47:0	**CTR**: **performance counter value**. Read-write,Volatile. Reset: 0000_0000_0000h.

MSRC001_0010 [System Configuration] (Core::X86::Msr::SYS_CFG)

Reset: 0000_0000_0000_0000h.	
_lthree[1:0]_core[3:0]; MSRC001_0010	
Bits	**Description**
63:24	Reserved.
23	**SMEE**: **secure memory encryption enable**. Reset: 0. 0=Memory encryption features are disabled. 1=Memory encryption features are enabled. Once Core::X86::Msr::HWCR[SmmLock] is set, this bit cannot be cleared until a reset. For enabling secure memory encryption see 2.1.4 [Memory Encryption].
	AccessType: Core::X86::Msr::HWCR_thread0[SmmLock] ? Read,Write-1-only : Read-write.
22	**Tom2ForceMemTypeWB**: **top of memory 2 memory type write back**. Read-write. Reset: 0. 1=The default memory type of memory between 4GB and Core::X86::Msr::TOM2 is write back instead of the memory type defined by Core::X86::Msr::MTRRdefType[MemType]. For this bit to have any effect, Core::X86::Msr::MTRRdefType[MtrrDefTypeEn] must be 1. MTRRs and PAT can be used to override this memory type.
21	**MtrrTom2En**: **MTRR top of memory 2 enable**. Read-write. Reset: 0. 0=Core::X86::Msr::TOM2 is disabled. 1= Core::X86::Msr::TOM2 is enabled.
20	**MtrrVarDramEn**: **MTRR variable DRAM enable**. Read-write. Reset: 0. Init: BIOS,1. 0=Core::X86::Msr::TOP_MEM and IORRs are disabled. 1=These registers are enabled.
19	**MtrrFixDramModEn**: **MTRR fixed RdDram and WrDram modification enable**. Read-write. Reset: 0. Check: 0. 0=Core::X86::Msr::MtrrFix_64K through Core::X86::Msr::MtrrFix_4K_7 [RdDram,WrDram] read values is masked 00b; writing does not change the hidden value. 1=Core::X86::Msr::MtrrFix_64K through Core::X86::Msr::MtrrFix_4K_7 [RdDram,WrDram] access type is Read-write. Not shared between threads. Controls access to Core::X86::Msr::MtrrFix_64K through Core::X86::Msr::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	**MtrrFixDramEn**: **MTRR fixed RdDram and WrDram attributes enable**. Read-write. Reset: 0. Init: BIOS,1. 1=Enables the RdDram and WrDram attributes in Core::X86::Msr::MtrrFix_64K through Core::X86::Msr::MtrrFix_4K_7.
17:0	Reserved.

MSRC001_0015 [Hardware Configuration] (Core::X86::Msr::HWCR)

Reset: 0000_0000_0100_0010h.	
_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0015	
Bits	**Description**
63:31	Reserved.
30	**IRPerfEn**: **enable instructions retired counter**. Read-write. Reset: 0. 1=Enable Core::X86::Msr::IRPerfCount.
29:28	Reserved.
27	**EffFreqReadOnlyLock**: **read-only effective frequency counter lock**. 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	**EffFreqCntMwait**: **effective frequency counting during mwait**. 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 2.1.5 [Effective Frequency].

25	**CpbDis**: **core performance boost disable**. 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	**TscFreqSel**: **TSC frequency select**. Read-only. Reset: 1. 1=The TSC increments at the P0 frequency.
23:22	Reserved.
21	**LockTscToCurrentP0**: **lock the TSC to the current P0 frequency**. 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	**IoCfgGpFault**: **IO-space configuration causes a GP fault**. 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	**McStatusWrEn**: **machine check status write enable**. Read-write. Reset: 0. 0=MCi_STATUS registers are readable; writing a non-zero pattern to these registers causes a general protection fault. 1=MCi_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.
	Description: McStatusWrEn can be used to debug machine check exception and interrupt handlers. See 3.1 [Machine Check Architecture].
17	**Wrap32Dis**: **32-bit address wrap disable**. 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 ±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	**RsmSpCycDis**: **RSM special bus cycle disable**. Reset: 0. 0=A link special bus cycle, SMIACK, is generated on a resume from SMI.
	AccessType: Core::X86::Msr::HWCR[SmmLock] ? Read-only : Read-write.
13	**SmiSpCycDis**: **SMI special bus cycle disable**. Reset: 0. 0=A link special bus cycle, SMIACK, is generated when an SMI interrupt is taken.
	AccessType: Core::X86::Msr::HWCR[SmmLock] ? Read-only : Read-write.
12:11	Reserved.
10	**MonMwaitUserEn**: **MONITOR/MWAIT user mode enable**. 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	**MonMwaitDis**: **MONITOR and MWAIT disable**. Read-write. Reset: 0. 1=The MONITOR and MWAIT opcodes become invalid. This affects what is reported back through Core::X86::Cpuid::FeatureIdEcx[Monitor].
8	**IgnneEm**: **IGNNE port emulation enable**. Read-write. Reset: 0. 1=Enable emulation of IGNNE port.
7	**AllowFerrOnNe**: **allow FERR on NE**. Read-write. Reset: 0. 0=Disable legacy FERR signaling and generate FERR exception directly. 1=Legacy FERR signaling.
6:5	Reserved.
4	**INVDWBINVD**: **INVD to WBINVD conversion**. Read-write. Reset: 1. Check: 1. 1=Convert INVD to WBINVD.
	Description: This bit is required to be set for normal operation when any of the following are true: • An L2 is shared by multiple threads.

	• An L3 is shared by multiple cores. • CC6 is enabled. • Probe filter is enabled.
3	**TlbCacheDis**: **cacheable memory disable**. Read-write. Reset: 0. 1=Disable performance improvement that assumes that the PML4, PDP, PDE and PTE entries are in cacheable WB DRAM.
	Description: Operating systems that maintain page tables in any other memory type must set the TlbCacheDis bit to insure proper operation. • TlbCacheDis does not override the memory type specified by the SMM ASeg and TSeg memory regions controlled by Core::X86::Msr::SMMAddr Core::X86::Msr::SMMMask.
2:1	Reserved.
0	**SmmLock**: **SMM code lock**. Read,Write-1-only. Reset: 0. Init: BIOS,1. Check: 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 2.1.12.1.10 [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:
CPUAddr[47:12] & PhyMask[47:12] == PhyBase[47:12] & 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.

_lthree[1:0]_core[3:0]_n0; MSRC001_0016
_lthree[1:0]_core[3:0]_n1; MSRC001_0018

Bits	Description
63:48	Reserved.
47:12	**PhyBase**: **physical base address**. Read-write. Reset: X_XXXX_XXXXh.
11:5	Reserved.
4	**RdMem**: **read from memory**. 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	**WrMem**: **write to memory**. 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.

_lthree[1:0]_core[3:0]_n0; MSRC001_0017
_lthree[1:0]_core[3:0]_n1; MSRC001_0019

Bits	Description
63:48	Reserved.
47:12	**PhyMask**: **physical address mask**. Read-write. Reset: 0_0000_0000h.
11	**Valid**. 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.

_lthree[1:0]_core[3:0]; MSRC001_001A

Bits	Description
63:48	Reserved.
47:23	**TOM[47:23]**: **top of memory**. 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 2.1.6.3 [System Address Map].
22:0	Reserved.

MSRC001_001D [Top Of Memory 2] (Core::X86::Msr::TOM2)

Read-write.

_lthree[1:0]_core[3:0]; MSRC001_001D

Bits	Description
63:48	Reserved.
47:23	**TOM2[47:23]**: **second top of memory**. 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 2.1.6.3 [System Address Map]. This register is enabled by Core::X86::Msr::SYS_CFG[MtrrTom2En].
22:0	Reserved.

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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0022

Bits	Description
63:10	Reserved.
9	**RedirSmiEn**. 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	**RedirVecEn**. 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	**RedirVector**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_n0; MSRC001_0030
_lthree[1:0]_core[3:0]_thread[1:0]_n1; MSRC001_0031
_lthree[1:0]_core[3:0]_thread[1:0]_n2; MSRC001_0032
_lthree[1:0]_core[3:0]_thread[1:0]_n3; MSRC001_0033
_lthree[1:0]_core[3:0]_thread[1:0]_n4; MSRC001_0034
_lthree[1:0]_core[3:0]_thread[1:0]_n5; MSRC001_0035

Bits	Description
63:56	**CpuNameString7**. Read-write. Reset: XXh.
55:48	**CpuNameString6**. Read-write. Reset: XXh.
47:40	**CpuNameString5**. Read-write. Reset: XXh.
39:32	**CpuNameString4**. Read-write. Reset: XXh.
31:24	**CpuNameString3**. Read-write. Reset: XXh.
23:16	**CpuNameString2**. Read-write. Reset: XXh.
15:8	**CpuNameString1**. Read-write. Reset: XXh.
7:0	**CpuNameString0**. Read-write. Reset: XXh.

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 a 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.	
_lthree[1:0]_core[3:0]_thread[1:0]_n0; MSRC001_0050	
_lthree[1:0]_core[3:0]_thread[1:0]_n1; MSRC001_0051	
_lthree[1:0]_core[3:0]_thread[1:0]_n2; MSRC001_0052	
_lthree[1:0]_core[3:0]_thread[1:0]_n3; MSRC001_0053	
Bits	**Description**
63	**SmiOnRdEn**: **enable SMI on IO read**. Read-write. Reset: 0. 1=Enables SMI generation on a read access.
62	**SmiOnWrEn**: **enable SMI on IO write**. Read-write. Reset: 0. 1=Enables SMI generation on a write access.
61	**ConfigSmi**: **configuration space SMI**. Read-write. Reset: 0. 0=IO access (that is not an IO-space configuration access). 1=Configuration access.
60:56	Reserved.
55:32	**SmiMask[23:0]**. Read-write. Reset: 00_0000h. 1=Do not mask address bit. 0=Mask address bit. SMI IO trap mask.
31:0	**SmiAddr[31:0]**. Read-write. Reset: 0000_0000h. SMI IO trap address.

MSRC001_0054 [IO Trap Control] (Core::X86::Msr::SMI_ON_IO_TRAP_CTL_STS)	
_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0054	
Bits	**Description**
63:32	Reserved. Read-only. Reset: Fixed,0000_0000h.
31:16	Reserved.
15	**IoTrapEn**: **IO trap enable**. 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	**SmiEn3**. Read-write. Reset: 0. 1=The trap Core::X86::Msr::SMI_ON_IO_TRAP_n[3] is enabled.
6	Reserved.
5	**SmiEn2**. Read-write. Reset: 0. 1=The trap Core::X86::Msr::SMI_ON_IO_TRAP_n[2] is enabled.
4	Reserved.
3	**SmiEn1**. Read-write. Reset: 0. 1=The trap Core::X86::Msr::SMI_ON_IO_TRAP_n[1] is enabled.
2	Reserved.
1	**SmiEn0**. 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.
_lthree[1:0]_core[3:0]; MSRC001_0055

Bits	Description
63:0	Reserved. Read-only. Reset: Fixed,0000_0000_0000_0000h.

MSRC001_0056 [SMI Trigger IO Cycle] (Core::X86::Msr::SmiTrigIoCycle)

Read-write. Reset: 0000_0000_0000_0000h.

See 2.1.12.1.3 [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.

_lthree[1:0]_core[3:0]_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.
15:0	**IoPortAddress**. Read-write. Reset: 0000h.

MSRC001_0058 [MMIO Configuration Base Address] (Core::X86::Msr::MmioCfgBaseAddr)

See 2.1.7 [Configuration Space] for a description of MMIO configuration space.

_lthree[1:0]_core[3:0]; MSRC001_0058

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:48</td><td>Reserved. Read-only. Reset: Fixed,0000h.</td></tr>
<tr><td>47:20</td><td><b>MmioCfgBaseAddr[47:20]</b>: <b>MMIO configuration base address bits[47:20]</b>. Read-write. Reset: XXX_XXXXh. Specifies the base address of the MMIO configuration range.</td></tr>
<tr><td>19:6</td><td>Reserved. Read-only. Reset: Fixed,0000h.</td></tr>
<tr><td>5:2</td><td><b>BusRange</b>: <b>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>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>1</td></tr>
<tr><td>1h</td><td>2</td></tr>
<tr><td>2h</td><td>4</td></tr>
<tr><td>3h</td><td>8</td></tr>
<tr><td>4h</td><td>16</td></tr>
<tr><td>5h</td><td>32</td></tr>
<tr><td>6h</td><td>64</td></tr>
<tr><td>7h</td><td>128</td></tr>
<tr><td>8h</td><td>256</td></tr>
<tr><td>Fh-9h</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>1</td><td>Reserved.</td></tr>
<tr><td>0</td><td><b>Enable</b>. Read-write. Reset: 0. 1=MMIO configuration space is enabled.</td></tr>
</table>

MSRC001_0061 [P-state Current Limit] (Core::X86::Msr::PStateCurLim)

_lthree[1:0]_core[3:0]; MSRC001_0061

Bits	Description
63:7	Reserved.
6:4	**PstateMaxVal**: **P-state maximum value**. 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. Read-only. Reset: Fixed,0.
2:0	**CurPstateLimit**: **current P-state limit**. 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)

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0062

Bits	Description
63:3	Reserved. Read,Error-on-write-1.
2:0	**PstateCmd**: **P-state change command**. 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.

_lthree[1:0]_core[3:0]; MSRC001_0063

Bits	Description
63:3	Reserved.
2:0	**CurPstate**: **current P-state**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_n0; MSRC001_0064
_lthree[1:0]_core[3:0]_thread[1:0]_n1; MSRC001_0065
_lthree[1:0]_core[3:0]_thread[1:0]_n2; MSRC001_0066
_lthree[1:0]_core[3:0]_thread[1:0]_n3; MSRC001_0067
_lthree[1:0]_core[3:0]_thread[1:0]_n4; MSRC001_0068
_lthree[1:0]_core[3:0]_thread[1:0]_n5; MSRC001_0069
_lthree[1:0]_core[3:0]_thread[1:0]_n6; MSRC001_006A
_lthree[1:0]_core[3:0]_thread[1:0]_n7; MSRC001_006B

Bits	Description
63	**PstateEn**. 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.
62:32	Reserved.
31:30	**IddDiv**: **current divisor**. Read-write. Reset: XXb. See IddValue.
29:22	**IddValue**: **current value**. 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	**CpuVid[7:0]**: **core VID**. Read-write. Reset: XXXXXXXXb.

13:8	**CpuDfsId**: **core divisor ID**. Read-write. Reset: XXXXXXb. Specifies the core frequency divisor; see CpuFid. For values [1Ah:08h], 1/8th integer divide steps supported down to VCO/3.25 (Note, L3/L2 fifo logic related to 4-cycle data heads-up requires core to be 1/3 of L3 frequency or higher). For values [30h:1Ch], 1/4th integer divide steps supported down to VCO/6 (DID[0] should zero if DID[5:0]>1Ah). (Note, core and L3 frequencies below 400MHz are not supported by the architecture). Core supports DID up to 30h, but L3 must be 2Ch (VCO/5.5) or less.

ValidValues:

Value	Description
00h	Off
07h-01h	Reserved.
08h	VCO/1
09h	VCO/1.125
1Ah-0Ah	VCO/<Value/8>
1Bh	Reserved.
1Ch	VCO/<Value/8>
1Dh	Reserved.
1Eh	VCO/<Value/8>
1Fh	Reserved.
20h	VCO/<Value/8>
21h	Reserved.
22h	VCO/<Value/8>
23h	Reserved.
24h	VCO/<Value/8>
25h	Reserved.
26h	VCO/<Value/8>
27h	Reserved.
28h	VCO/<Value/8>
29h	Reserved.
2Ah	VCO/<Value/8>
2Bh	Reserved.
2Ch	VCO/<Value/8>
3Fh-2Dh	Reserved.

7:0	**CpuFid[7:0]**: **core frequency ID**. Read-write. Reset: XXh. Specifies the core frequency multiplier. The core COF is a function of CpuFid and CpuDid, and defined by CoreCOF.

ValidValues:

Value	Description
0Fh-00h	Reserved.
FFh-10h	<Value>*25

MSRC001_0073 [C-state Base Address] (Core::X86::Msr::CStateBaseAddr)

Read-write. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0073

Bits	Description
63:16	Reserved.
15:0	**CstateAddr**: **C-state address**. 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_0074 [CPU Watchdog Timer] (Core::X86::Msr::CpuWdtCfg)

Read-write. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]; MSRC001_0074

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:7</td><td>Reserved.</td></tr>
<tr><td>6:3</td><td><b>CpuWdtCountSel</b>: <b>CPU watchdog timer count select</b>. Read-write. Reset: 0h. CpuWdtCountSel and CpuWdtTimeBase together specify the time period required for the WDT to expire. The time period is ((the multiplier specified by CpuWdtCountSel) * (the time base specified by CpuWdtTimeBase)). The actual timeout period may be anywhere from zero to one increment less than the values specified, due to non-deterministic behavior.</td></tr>
<tr><td></td><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>4095</td></tr>
<tr><td>1h</td><td>2047</td></tr>
<tr><td>2h</td><td>1023</td></tr>
<tr><td>3h</td><td>511</td></tr>
<tr><td>4h</td><td>255</td></tr>
<tr><td>5h</td><td>127</td></tr>
<tr><td>6h</td><td>63</td></tr>
<tr><td>7h</td><td>31</td></tr>
<tr><td>8h</td><td>8191</td></tr>
<tr><td>9h</td><td>16383</td></tr>
<tr><td>Fh-Ah</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>2:1</td><td><b>CpuWdtTimeBase</b>: <b>CPU watchdog timer time base</b>. Read-write. Reset: 0h. Specifies the time base for the timeout period specified in CpuWdtCountSel.</td></tr>
<tr><td></td><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>1.31ms</td></tr>
<tr><td>1h</td><td>1.28us</td></tr>
<tr><td>3h-2h</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>0</td><td><b>CpuWdtEn</b>: <b>CPU watchdog timer enable</b>. Read-write. Reset: 0. Init: BIOS,1. 1=The WDT is enabled.</td></tr>
</table>

MSRC001_0111 [SMM Base Address] (Core::X86::Msr::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 2.1.12.1.5 [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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0111

Bits	Description
63:32	Reserved.
31:0	**SmmBase**. Reset: 0003_0000h.
	AccessType: Core::X86::Msr::HWCR[SmmLock] ? Read-only : Read-write.

MSRC001_0112 [SMM TSeg Base Address] (Core::X86::Msr::SMMAddr)

Configurable. Reset: 0000_0000_0000_0000h.

See 2.1.12.1 [System Management Mode (SMM)] and 2.1.6.3.1 [Memory Access to the Physical Address Space]. See Core::X86::Msr::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:

CPUAddr[47:17] & TSegMask[47:17] == TSegBase[47:17] & TSegMask[47:17].

For example, if TSeg spans 256 KB and starts at the 1 MB address. The Core::X86::Msr::SMMAddr[TSegBase[47:17]] would be set to 0010_0000h and the Core::X86::Msr::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.

_lthree[1:0]_core[3:0]; MSRC001_0112

Bits	Description
63:48	Reserved.
47:17	**TSegBase[47:17]**: **TSeg address range base**. Configurable. Reset: 0000_0000h. AccessType: (Core::X86::Msr::HWCR[SmmLock]) ? Read-only : Read-write.
16:0	Reserved.

MSRC001_0113 [SMM TSeg Mask] (Core::X86::Msr::SMMMask)

Configurable. Reset: 0000_0000_0000_0000h.

See 2.1.12.1 [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::Msr::SMMAddr[TSegBase[47:17]]) with a variable size (specified by Core::X86::Msr::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 2.1.6.3.1.1 [Determining Memory Type].

_lthree[1:0]_core[3:0]; MSRC001_0113

Bits	Description
63:48	Reserved.
47:17	**TSegMask[47:17]**: **TSeg address range mask**. Configurable. Reset: 0000_0000h. See Core::X86::Msr::SMMAddr. AccessType: (Core::X86::Msr::HWCR[SmmLock]) ? Read-only : Read-write.
16:15	Reserved.
14:12	**TMTypeDram**: **TSeg address range memory type**. Configurable. Reset: 0h. Specifies the memory type for SMM accesses to the TSeg range that are directed to DRAM. AccessType: (Core::X86::Msr::HWCR[SmmLock]) ? Read-only : Read-write.

ValidValues:

Value	Description
0h	UC or uncacheable.
1h	WC or write combining.

<table>
<tr><td></td><td><table>
<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></td></tr>
<tr><td>11</td><td>Reserved.</td></tr>
<tr><td>10:8</td><td><b>AMTypeDram</b>: <b>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::Msr::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></td></tr>
<tr><td>7:6</td><td>Reserved.</td></tr>
<tr><td>5</td><td><b>TMTypeIoWc</b>: <b>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::Msr::HWCR[SmmLock]) ? Read-only : Read-write.</td></tr>
<tr><td>4</td><td><b>AMTypeIoWc</b>: <b>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::Msr::HWCR[SmmLock]) ? Read-only : Read-write.</td></tr>
<tr><td>3</td><td><b>TClose</b>: <b>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::Msr::HWCR[SmmLock]) ? Read-only : Read-write.</td></tr>
<tr><td>2</td><td><b>AClose</b>: <b>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::Msr::HWCR[SmmLock]) ? Read-only : Read-write.</td></tr>
<tr><td>1</td><td><b>TValid</b>: <b>enable TSeg SMM address range</b>. Configurable. Reset: 0. 1=The TSeg address range SMM enabled. AccessType: (Core::X86::Msr::HWCR[SmmLock]) ? Read-only : Read-write.</td></tr>
<tr><td>0</td><td><b>AValid</b>: <b>enable ASeg SMM address range</b>. Configurable. Reset: 0. 1=The ASeg address range SMM enabled. AccessType: (Core::X86::Msr::HWCR[SmmLock]) ? Read-only : Read-write.</td></tr>
</table>

MSRC001_0114 [Virtual Machine Control] (Core::X86::Msr::VM_CR)

Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0114

Bits	Description
63:32	Reserved.
31:5	Reserved. Read-only,Error-on-write-1. Reset: 000_0000h.
4	**SvmeDisable**: **SVME disable**. Configurable. Reset: 0. 0=Core::X86::Msr::EFER[SVME] is read-write. 1=Core::X86::Msr::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::Msr::EFER[SVME]==1) causes a #GP fault, regardless of the state of Lock. See the APM2 section titled "Enabling SVM" for software use of this field.
3	**Lock**: **SVM lock**. Read-only,Volatile. Reset: 0. 0=SvmeDisable is read-write. 1=SvmeDisable is read-only. See Core::X86::Msr::SvmLockKey[SvmLockKey] for the condition that causes hardware to clear this field.

2	Reserved.
1	**InterceptInit**: **intercept INIT**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0115

Bits	Description
63:32	Reserved.
31:1	Reserved. Read-only,Error-on-write-1. Reset: 0000_0000h.
0	**IGNNE**: **current IGNNE state**. Read-write. Reset: 0. This bit controls the current state of the processor internal IGNNE signal.

MSRC001_0116 [SMM Control] (Core::X86::Msr::SMM_CTL)

Reset: 0000_0000_0000_0000h.

The bits in this register are processed in the order of: SmmEnter, SmiCycle, SmmDismiss, RsmCycle and SmmExit. However, only the following combination of bits may be set in a single write (all other combinations result in undefined behavior):

- SmmEnter and SmiCycle.
- SmmEnter and SmmDismiss.
- SmmEnter, SmiCycle and SmmDismiss.
- SmmExit and RsmCycle.

Software is responsible for ensuring that SmmEnter and SmmExit operations are properly matched and are not nested.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0116

Bits	Description
63:5	Reserved. Error-on-read,Error-on-write-1. Reset: 000_0000_0000_0000h.
4	**RsmCycle**: **send RSM special cycle**. Reset: 0. 1=Send a RSM special cycle.
	AccessType: Core::X86::Msr::HWCR[SmmLock] ? Error-on-read,Error-on-write : Write-only,Error-on-read.
3	**SmmExit**: **exit SMM**. Reset: 0. 1=Exit SMM.
	AccessType: Core::X86::Msr::HWCR[SmmLock] ? Error-on-read,Error-on-write : Write-only,Error-on-read.
2	**SmiCycle**: **send SMI special cycle**. Reset: 0. 1=Send a SMI special cycle.
	AccessType: Core::X86::Msr::HWCR[SmmLock] ? Error-on-read,Error-on-write : Write-only,Error-on-read.
1	**SmmEnter**: **enter SMM**. Reset: 0. 1=Enter SMM.
	AccessType: Core::X86::Msr::HWCR[SmmLock] ? Error-on-read,Error-on-write : Write-only,Error-on-read.
0	**SmmDismiss**: **clear SMI**. Reset: 0. 1=Clear the SMI pending flag.
	AccessType: Core::X86::Msr::HWCR[SmmLock] ? Error-on-read,Error-on-write : Write-only,Error-on-read.

MSRC001_0117 [Virtual Machine Host Save Physical Address] (Core::X86::Msr::VM_HSAVE_PA)

Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0117

Bits	Description
63:48	Reserved. Read-only,Error-on-write-1. Reset: 0000h.
47:12	**VM_HSAVE_PA**: **physical address of host save area**. 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. Read-only,Error-on-write-1. Reset: 000h.

MSRC001_0118 [SVM Lock Key] (Core::X86::Msr::SvmLockKey)

Read-write. Reset: Fixed,0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0118

Bits	Description
63:0	**SvmLockKey**: **SVM lock key**. 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.

MSRC001_011A [Local SMI Status] (Core::X86::Msr::LocalSmiStatus)

Read-write. Reset: 0000_0000_0000_0000h.

This register returns the same information that is returned in Core::X86::Smm::LocalSmiStatus portion of the SMM save state. The information in this register is only updated when Core::X86::Msr::SMM_CTL[SmmDismiss] is set by software.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_011A

Bits	Description
63:32	Reserved.
31:0	**LocalSmiStatus**. Read-write. Reset: 0000_0000h. See Core::X86::Smm::LocalSmiStatus.

MSRC001_011B [AVIC Doorbell] (Core::X86::Msr::AvicDoorbell)

Reset: 0000_0000_0000_0000h.

The ApicId is a physical APIC Id; not valid for logical APIC ID.
See Core::X86::Cpuid::SvmRevFeatIdEdx[AVIC].

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_011B

Bits	Description
63:8	Reserved. Error-on-read,Error-on-write-1. Reset: 00_0000_0000_0000h.
7:0	**ApicId**: **APIC ID [7:0]**. Write-only,Error-on-read. Reset: 00h.

MSRC001_011E [VM Page Flush] (Core::X86::Msr::VMPAGE_FLUSH)

Writes to this MSR cause 4KB of encrypted, guest-tagged data to be flushed from caches if present.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_011E

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td rowspan="2">63:12</td><td><b>VirtualAddr</b>. Reset: X_XXXX_XXXX_XXXXh. Guest physical address of page to flush.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[SMEE] ? Write-only,Error-on-read : Error-on-read,Error-on-write.</td></tr>
<tr><td rowspan="3">11:0</td><td><b>ASID</b>. Reset: XXXh. ASID to use for flush. Writing reserved values generates #GP.</td></tr>
<tr><td>AccessType: Core::X86::Msr::SYS_CFG[SMEE] ? Write-only,Error-on-read : Error-on-read,Error-on-write.</td></tr>
<tr><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>000h</td><td>Reserved.</td></tr>
<tr><td>00Fh-001h</td><td>Valid ASID</td></tr>
<tr><td>FFFh-010h</td><td>Reserved.</td></tr>
</table>
</td></tr>
</table>

MSRC001_0130 [Guest Host Communication Block] (Core::X86::Msr::GHCB)

Read-write. Reset: 0000_0000_0000_0000h.

If Core::X86::Msr::GHCB is accessed in hypervisor mode, #GP is generated.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0130

Bits	Description
63:0	**GHCBPA**. Read-write. Reset: 0000_0000_0000_0000h. Guest physical address of GHCB.

MSRC001_0131 [SEV Status] (Core::X86::Msr::SEV_Status)

Read,Error-on-write. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0131

Bits	Description
63:2	Reserved.
1	**SevEsEnabled**. Read,Error-on-write. Reset: 0. 1=The guest was launched with the Sev-ES feature enabled in VMCB offset 90h.
0	**SevEnabled**. Read,Error-on-write. Reset: 0. 1=The guest was launched with SEV feature enabled in VMCB offset 90h.

MSRC001_0140 [OS Visible Work-around Length] (Core::X86::Msr::OSVW_ID_Length)

Read-write. Reset: 0000_0000_0000_0000h.	
_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0140	
Bits	**Description**
63:16	Reserved.
15:0	**OSVWIdLength**: **OS visible work-around ID length**. 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.	
_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_0141	
Bits	**Description**
63:0	**OsvwStatusBits**: **OS visible work-around status bits**. Read-write. Reset: 0000_0000_0000_0000h. See the Revision Guide for the definition of this field; see 1.2 [Reference Documents].

MSRC001_020[0...A] [Performance Event Select [5:0]] (Core::X86::Msr::PERF_CTL)

<table>
<tr><td colspan="2">Read-write. Reset: 0000_0000_0000_0000h.</td></tr>
<tr><td colspan="2">See 2.1.15 [Performance Monitor Counters]. Core::X86::Msr::PERF_LEGACY_CTL is an alias of MSRC001_020[6,4,2,0].</td></tr>
<tr><td colspan="2">_lthree[1:0]_core[3:0]_thread[1:0]_n0; MSRC001_0200</td></tr>
<tr><td colspan="2">_lthree[1:0]_core[3:0]_thread[1:0]_n1; MSRC001_0202</td></tr>
<tr><td colspan="2">_lthree[1:0]_core[3:0]_thread[1:0]_n2; MSRC001_0204</td></tr>
<tr><td colspan="2">_lthree[1:0]_core[3:0]_thread[1:0]_n3; MSRC001_0206</td></tr>
<tr><td colspan="2">_lthree[1:0]_core[3:0]_thread[1:0]_n4; MSRC001_0208</td></tr>
<tr><td colspan="2">_lthree[1:0]_core[3:0]_thread[1:0]_n5; MSRC001_020A</td></tr>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:42</td><td>Reserved.</td></tr>
<tr><td>41:40</td><td><b>HostGuestOnly</b>: <b>count only host/guest events</b>. Read-write. Reset: 0h.</td></tr>
<tr><td></td><td><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></td></tr>
<tr><td>39:36</td><td>Reserved.</td></tr>
<tr><td>35:32</td><td><b>EventSelect[11:8]</b>: <b>performance event select</b>. Read-write. Reset: 0h.</td></tr>
<tr><td>31:24</td><td><b>CntMask</b>: <b>counter mask</b>. Read-write. Reset: 00h. Controls the number of events counted per clock cycle.</td></tr>
<tr><td></td><td><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. Maximum number of events in one cycle is 32.</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>
</table></td></tr>
</table>

FFh-80h	Reserved.

23	**Inv**: **invert counter mask**. Read-write. Reset: 0. See CntMask.
22	**En**: **enable performance counter**. Read-write. Reset: 0. 1=Performance event counter is enabled.
21	Reserved.
20	**Int**: **enable APIC interrupt**. 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.
19	Reserved.
18	**Edge**: **edge detect**. Read-write. Reset: 0. 0=Level detect. 1=Zero-to-one Edge detect. Read-write. 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	**OsUserMode**: **OS and user mode**. Read-write. Reset: 0h.
	ValidValues:

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	**UnitMask**: **event qualification**. 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	**EventSelect[7:0]**: **event select**. 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 2.1.15.4 [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)

See Core::X86::Msr::PERF_CTL. 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].

_lthree[1:0]_core[3:0]_thread[1:0]_n0; MSRC001_0201
_lthree[1:0]_core[3:0]_thread[1:0]_n1; MSRC001_0203
_lthree[1:0]_core[3:0]_thread[1:0]_n2; MSRC001_0205
_lthree[1:0]_core[3:0]_thread[1:0]_n3; MSRC001_0207
_lthree[1:0]_core[3:0]_thread[1:0]_n4; MSRC001_0209
_lthree[1:0]_core[3:0]_thread[1:0]_n5; MSRC001_020B

Bits	Description
63:48	Reserved. Read-only. Reset: Fixed,0000h.
47:0	**CTR**: **performance counter value**. Read-write,Volatile. Reset: 0000_0000_0000h.

MSRC001_023[0...A] [L3 Performance Event Select [5:0]] (Core::X86::Msr::ChL3PmcCfg)

Read-write. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_n0; MSRC001_0230
_lthree[1:0]_n1; MSRC001_0232
_lthree[1:0]_n2; MSRC001_0234

_lthree[1:0]_n3; MSRC001_0236	
_lthree[1:0]_n4; MSRC001_0238	
_lthree[1:0]_n5; MSRC001_023A	
Bits	**Description**
63:56	**ThreadMask**. Read-write. Reset: 00h. Controls which of the 8 threads in the complex are being counted. One or more threads must be selected.
	ValidValues:

Bit	Description
[0]	Core 0 Thread 0 mask.
[1]	Core 0 Thread 1 mask.
[2]	Core 1 Thread 0 mask.
[3]	Core 1 Thread 1mask.
[4]	Core 2 Thread 0 mask.
[5]	Core 2 Thread 1 mask.
[6]	Core 3 Thread 0 mask.
[7]	Core 3 Thread 1 mask.

Bits	Description
55:52	Reserved.
51:48	**SliceMask**. Read-write. Reset: 0h. Controls which L3 slices are counting this event. One or more Slices must be selected.
	ValidValues:

Bit	Description
[0]	L3 Slice 0 mask.
[1]	L3 Slice 1 mask.
[2]	L3 Slice 2 mask.
[3]	L3 Slice 3 mask.

Bits	Description
47:23	Reserved.
22	**Enable**: **Enable L3 performance counter**. Read-write. Reset: 0. 1=Enable.
21:16	Reserved.
15:8	**UnitMask**: **event qualification**. 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	**EventSel**: **event select**. Read-write. Reset: 00h.

MSRC001_023[1...B] [L3 Performance Event Counter [5:0]] (Core::X86::Msr::ChL3Pmc)

Reset: 0000_0000_0000_0000h.	
Also can be read via x86 instructions RDPMC ECX=[0F:0A].	
_lthree[1:0]_n0; MSRC001_0231	
_lthree[1:0]_n1; MSRC001_0233	
_lthree[1:0]_n2; MSRC001_0235	
_lthree[1:0]_n3; MSRC001_0237	
_lthree[1:0]_n4; MSRC001_0239	
_lthree[1:0]_n5; MSRC001_023B	
Bits	**Description**
63:49	Reserved.
48	**Overflow**. Read-write. Reset: 0.
47:32	**CountHi**. Read-write,Volatile. Reset: 0000h.
31:0	**CountLo**. Read-write,Volatile. Reset: 0000_0000h.

MSRC001_024[0...6] [Data Fabric Performance Event Select [3:0]] (Core::X86::Msr::DF_PERF_CTL)

Read-write. Reset: 0000_0000_0000_0000h.

See 2.1.15 [Performance Monitor Counters].

Note: To get meaningful data, each of the counters should be similarly programmed across events selected.

_n0; MSRC001_0240
_n1; MSRC001_0242
_n2; MSRC001_0244
_n3; MSRC001_0246

Bits	Description
63:61	Reserved.
60:59	**EventSelect[13:12]**: **performance event select**. Read-write. Reset: 0h.
58:36	Reserved.
35:32	**EventSelect[11:8]**: **performance event select**. Read-write. Reset: 0h. See EventSelect[7:0].
31:23	Reserved.
22	**En**: **enable performance counter**. Read-write. Reset: 0. 1=Performance event counter is enabled.
21:16	Reserved.
15:8	**UnitMask**: **event qualification**. 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	**EventSelect[7:0]**: **event select**. Read-write. Reset: 00h. This field, along with EventSelect[13:12] and EventSelect[11:8] above, combine to form the 14-bit event select field, EventSelect[13:0]. 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_024[1...7] [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 RDPMC ECX=[09:06].

_n0; MSRC001_0241
_n1; MSRC001_0243
_n2; MSRC001_0245
_n3; MSRC001_0247

Bits	Description
63:48	Reserved. Read-only. Reset: Fixed,0000h.
47:0	**CTR[47:0]**: **performance counter value[47:0]**. Read-write,Volatile. Reset: 0000_0000_0000h. The current value of the event counter.

MSRC001_0299 [RAPL Power Unit] (Core::X86::Msr::RAPL_PWR_UNIT)

Read-only,Volatile. Reset: 0000_0000_000A_1003h.

_lthree[1:0]; MSRC001_0299

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:20</td><td>Reserved.</td></tr>
<tr><td rowspan="2">19:16</td><td><b>TU</b>: <b>Time Units in seconds</b>. Read-only,Volatile. Reset: Ah. Time information (in Seconds) is based on the multiplier, 1/ 2^TU; where TU is an unsigned integer. Default value is 1010b, indicating time unit is in 976 microseconds increment.</td></tr>
<tr><td><b>ValidValues</b>:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Fh-0h</td><td>1/2^<Value> Seconds</td></tr></table></td></tr>
<tr><td>15:13</td><td>Reserved.</td></tr>
</table>

<table>
<tr><td>12:8</td><td><b>ESU</b>: <b>Energy Status Units</b>. Read-only,Volatile. Reset: 10h. Energy information (in Joules) is based on the multiplier, 1/2^ESU; where ESU is an unsigned integer. Default value is 10000b, indicating energy status unit is in 15.3 micro-Joules increment.</td></tr>
<tr><td></td><td><b>ValidValues</b>:<table><tr><th>Value</th><th>Description</th></tr><tr><td>1Fh-00h</td><td>1/2^<Value> Joules</td></tr></table></td></tr>
<tr><td>7:4</td><td>Reserved.</td></tr>
<tr><td>3:0</td><td><b>PU</b>: <b>Power Units</b>. Read-only,Volatile. Reset: 3h. 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.</td></tr>
<tr><td></td><td><b>ValidValues</b>:<table><tr><th>Value</th><th>Description</th></tr><tr><td>Fh-0h</td><td>1/2^<Value> Watts</td></tr></table></td></tr>
</table>

MSRC001_029A [Core Energy Status] (Core::X86::Msr::CORE_ENERGY_STAT)

Read-only,Volatile. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]; MSRC001_029A

Bits	Description
63:32	Reserved.
31:0	**TotalEnergyConsumed**. Read-only,Volatile. Reset: 0000_0000h.

MSRC001_029B [Package Energy Status] (Core::X86::Msr::PKG_ENERGY_STAT)

Read-only,Volatile. Reset: 0000_0000_0000_0000h.

_lthree[1:0]; MSRC001_029B

Bits	Description
63:32	Reserved.
31:0	**TotalEnergyConsumed**. Read-only,Volatile. Reset: 0000_0000h.

2.1.14.4 MSRs - MSRC001_1xxx

See 1.3.3 [Register Mnemonics] for a description of the register naming convention. MSRs are accessed through x86 WRMSR and RDMSR instructions.

MSRC001_1002 [CPUID Features for CPUID Fn00000007_E[A,B]X] (Core::X86::Msr::CPUID_7_Features)

Read-write.

Core::X86::Msr::CPUID_7_Features[63:32] provides control over values read from Core::X86::Cpuid::StructExtFeatIdEax0; Core::X86::Msr::CPUID_7_Features[31:0] provides control over values read from Core::X86::Cpuid::StructExtFeatIdEbx0.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1002

Bits	Description
63:30	Reserved.
29	**SHA**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[SHA].
28:24	Reserved.
23	**CLFSHOPT**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[CLFSHOPT].
22:21	Reserved.
20	**SMAP**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[SMAP].
19	**ADX**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[ADX].
18	**RDSEED**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[RDSEED].
17:9	Reserved.

8	**BMI2**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[BMI2].
7	**SMEP**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[SMEP].
6	Reserved.
5	**AVX2**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[AVX2].
4	Reserved.
3	**BMI1**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[BMI1].
2:1	Reserved.
0	**FSGSBASE**. Read-write. Reset: Core::X86::Cpuid::StructExtFeatIdEbx0[FSGSBASE].

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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1003

Bits	Description
63:1	Reserved.
0	**EffFreq**. Read-write. Reset: 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1004

Bits	Description
63	Reserved.
62	**RDRAND**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[RDRAND].
61	**F16C**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[F16C].
60	**AVX**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[AVX].
59	**OSXSAVE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[OSXSAVE]. Modifies Core::X86::Cpuid::FeatureIdEcx[OSXSAVE] only if CR4[OSXSAVE].
58	**XSAVE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[XSAVE].
57	**AES**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[AES]. Modifies Core::X86::Cpuid::FeatureIdEcx[AES] only if the reset value is 1.
56	Reserved.
55	**POPCNT**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[POPCNT].
54	**MOVBE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[MOVBE].
53	Reserved.
52	**SSE42**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[SSE42].
51	**SSE41**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[SSE41].
50	Reserved.
49	**PCID**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[PCID].
48:46	Reserved.
45	**CMPXCHG16B**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[CMPXCHG16B].
44	**FMA**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[FMA].
43:42	Reserved.
41	**SSSE3**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[SSSE3].
40:36	Reserved.
35	**Monitor**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[Monitor]. Modifies Core::X86::Cpuid::FeatureIdEcx[Monitor] only if ~Core::X86::Msr::HWCR[MonMwaitDis].
34	Reserved.

33	**PCLMULQDQ**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[PCLMULQDQ]. Modifies Core::X86::Cpuid::FeatureIdEcx[PCLMULQDQ] only if the reset value is 1.
32	**SSE3**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEcx[SSE3].
31:29	Reserved.
28	**HTT**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[HTT].
27	Reserved.
26	**SSE2**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[SSE2].
25	**SSE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[SSE].
24	**FXSR**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[FXSR].
23	**MMX**: **MMX™ instructions**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MMX].
22:20	Reserved.
19	**CLFSH**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[CLFSH].
18	Reserved.
17	**PSE36**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PSE36].
16	**PAT**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PAT].
15	**CMOV**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[CMOV].
14	**MCA**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MCA].
13	**PGE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PGE].
12	**MTRR**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MTRR].
11	**SysEnterSysExit**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[SysEnterSysExit].
10	Reserved.
9	**APIC**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[APIC]. Modifies Core::X86::Cpuid::FeatureIdEdx[APIC] only if Core::X86::Msr::APIC_BAR[ApicEn].
8	**CMPXCHG8B**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[CMPXCHG8B].
7	**MCE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MCE].
6	**PAE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PAE].
5	**MSR**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[MSR].
4	**TSC**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[TSC].
3	**PSE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[PSE].
2	**DE**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[DE].
1	**VME**. Read-write. Reset: Core::X86::Cpuid::FeatureIdEdx[VME].
0	**FPU**. Read-write. Reset: 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1005

Bits	Description
63:62	Reserved.
61	**MwaitExtended**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[MwaitExtended].
60	**PerfCtrExtLLC**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[PerfCtrExtLLC].
59	**PerfTsc**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[PerfTsc].
58	**DataBreakpointExtension**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[DataBreakpointExtension].
57	Reserved.
56	**PerfCtrExtDF**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[PerfCtrExtDF].
55	**PerfCtrExtCore**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[PerfCtrExtCore].
54	**TopologyExtensions**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[TopologyExtensions].

53:50	Reserved.
49	**TCE**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[TCE].
48	**FMA4**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[FMA4]. Init: 0.
47	**LWP**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[LWP].
46	Reserved.
45	**WDT**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[WDT].
44	**SKINIT**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[SKINIT].
43	**XOP**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[XOP].
42	**IBS**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[IBS]. Init: BIOS,0. To enable the IBS feature, use BIOS setup option.
41	**OSVW**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[OSVW].
40	**ThreeDNowPrefetch**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[ThreeDNowPrefetch].
39	**MisAlignSse**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[MisAlignSse].
38	**SSE4A**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[SSE4A].
37	**ABM**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[ABM].
36	**AltMovCr8**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[AltMovCr8].
35	**ExtApicSpace**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[ExtApicSpace].
34	**SVM**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[SVM].
33	**CmpLegacy**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[CmpLegacy].
32	**LahfSahf**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEcx[LahfSahf].
31	**ThreeDNow**: **3DNow!™ instructions**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[ThreeDNow].
30	**ThreeDNowExt**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[ThreeDNowExt].
29	**LM**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[LM].
28	Reserved.
27	**RDTSCP**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[RDTSCP].
26	**Page1GB**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[Page1GB].
25	**FFXSR**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[FFXSR].
24	**FXSR**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[FXSR].
23	**MMX**: **MMX™ instructions**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[MMX].
22	**MmxExt**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[MmxExt].
21	Reserved.
20	**NX**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[NX].
19:18	Reserved.
17	**PSE36**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[PSE36].
16	**PAT**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[PAT].
15	**CMOV**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[CMOV].
14	**MCA**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[MCA].
13	**PGE**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[PGE].
12	**MTRR**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[MTRR].
11	**SysCallSysRet**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[SysCallSysRet].
10	Reserved.
9	**APIC**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[APIC].
8	**CMPXCHG8B**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[CMPXCHG8B].
7	**MCE**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[MCE].
6	**PAE**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[PAE].
5	**MSR**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[MSR].
4	**TSC**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[TSC].
3	**PSE**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[PSE].
2	**DE**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[DE].

1	**VME**. Read-write. Reset: Core::X86::Cpuid::FeatureExtIdEdx[VME].
0	**FPU**. Read-write. Reset: 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::FeatureExtIdEcx[DataBreakpointExtension].

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1019

Bits	Description
63:32	Reserved.
31:0	**AddrMask**: **mask for DR linear address data breakpoint DR1**. 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::FeatureExtIdEcx[DataBreakpointExtension].

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_101A

Bits	Description
63:32	Reserved.
31:0	**AddrMask**: **mask for DR linear address data breakpoint DR2**. 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::FeatureExtIdEcx[DataBreakpointExtension].

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_101B

Bits	Description
63:32	Reserved.
31:0	**AddrMask**: **mask for DR linear address data breakpoint DR3**. 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_1023 [Table Walker Configuration] (Core::X86::Msr::TW_CFG)

Read-write. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]; MSRC001_1023

Bits	Description
63:50	Reserved.
49	**TwCfgCombineCr0Cd**: **combine CR0_CD for both threads of a core**. Read-write. Reset: 0. Init: BIOS,1. 1=The host Cr0_Cd values from the two threads are OR'd together and used by both threads.
48:0	Reserved.

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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1027

Bits	Description
63:32	Reserved.

31:0	**DR0**: **mask for DR0 linear address data breakpoint**. 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 2.1.16 [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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1030

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>63:59</td><td>Reserved.</td></tr>
<tr><td>58</td><td><b>IbsFetchL2Miss</b>: <b>L2 cache miss for the sampled fetch</b>. Read-only,Volatile. Reset: 0. 1=The instruction fetch missed in the L2 Cache. Qualified by (IbsFetchComp == 1).</td></tr>
<tr><td>57</td><td><b>IbsRandEn</b>: <b>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.</td></tr>
<tr><td>56</td><td><b>IbsL2TlbMiss</b>: <b>instruction cache L2TLB miss</b>. Read-only,Volatile. Reset: 0. 1=The instruction fetch missed in the L2 TLB.</td></tr>
<tr><td>55</td><td><b>IbsL1TlbMiss</b>: <b>instruction cache L1TLB miss</b>. Read-only,Volatile. Reset: 0. 1=The instruction fetch missed in the L1 TLB.</td></tr>
<tr><td rowspan="2">54:53</td><td><b>IbsL1TlbPgSz</b>: <b>instruction cache L1TLB page size</b>. Read-only,Volatile. Reset: 0h. Indicates the page size of the translation in the L1 TLB. This field is only valid if IbsPhyAddrValid == 1.</td></tr>
<tr><td><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>Reserved.</td></tr>
</table></td></tr>
<tr><td>52</td><td><b>IbsPhyAddrValid</b>: <b>instruction fetch physical address valid</b>. Read-only,Volatile. Reset: 0. 1=The physical address in Core::X86::Msr::IBS_FETCH_PHYSADDR and the IbsL1TlbPgSz field are valid for the instruction fetch.</td></tr>
<tr><td>51</td><td><b>IbsIcMiss</b>: <b>instruction cache miss</b>. Read-only,Volatile. Reset: 0. 1=The instruction fetch missed in the instruction cache.</td></tr>
<tr><td>50</td><td><b>IbsFetchComp</b>: <b>instruction fetch complete</b>. Read-only,Volatile. Reset: 0. 1=The instruction fetch completed and</td></tr>
</table>

	the data is available for use by the instruction decoder.
49	**IbsFetchVal**: **instruction fetch valid**. Read-only,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::Msr::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	**IbsFetchEn**: **instruction fetch enable**. Read-write. Reset: 0. 1=Instruction fetch sampling is enabled.
47:32	**IbsFetchLat**: **instruction fetch latency**. Read-only,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	**IbsFetchCnt[19:4]**. 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	**IbsFetchMaxCnt[19:4]**. 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::Msr::IBS_FETCH_LINADDR)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1031

Bits	Description
63:0	**IbsFetchLinAd**: **instruction fetch linear address**. 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::Msr::IBS_FETCH_PHYSADDR)

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1032

Bits	Description
63:48	Reserved. Read-only. Reset: Fixed,0000h.
47:0	**IbsFetchPhysAd**: **instruction fetch physical address**. 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::Msr::IBS_FETCH_CTL[IbsPhyAddrValid] is asserted.

MSRC001_1033 [IBS Execution Control] (Core::X86::Msr::IBS_OP_CTL)

Reset: 0000_0000_0000_0000h.

See 2.1.16 [Instruction Based Sampling (IBS)].
The IBS execution sampling engine is described as follows for IbsOpCntCtl == 1. If IbsOpCntCtl == 1n 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 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 micro-op counter [26:0].
- When the periodic micro-op counter reaches IbsOpMaxCnt:
  - The next dispatched micro-op is tagged if IbsOpCntCtl == 1. A valid op in the next dispatched line is tagged if IbsOpCntCtl == 0. See IbsOpCntCtl.
  - The periodic micro-op counter [26:7]=0; [6:0] is randomized by hardware.
- The periodic micro-op counter is not modified when a tagged micro-op is flushed.
- When a tagged micro-op is retired:
  - 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1033

Bits	Description
63:59	Reserved.
58:32	**IbsOpCurCnt[26:0]**: **periodic op counter current count**. Read-write,Volatile. Reset: 000_0000h. Returns the current value of the periodic op counter.
31:27	Reserved.
26:20	**IbsOpMaxCnt[26:20]**: **periodic op counter maximum count**. Read-write. Reset: 00h. See IbsOpMaxCnt[19:4].
19	**IbsOpCntCtl**: **periodic op counter count control**. 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 Micro-Ops; when a roll-over occurs, the counter is preloaded with a pseudorandom 7 bit value between 1 and 127.
18	**IbsOpVal**: **micro-op sample valid**. 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	**IbsOpEn**: **micro-op sampling enable**. Read-write. Reset: 0. 1=Instruction execution sampling enabled.
16	Reserved.
15:0	**IbsOpMaxCnt[19:4]**: **periodic op counter maximum count**. 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.
	ValidValues:

Value	Description
0008h-0000h	Reserved.
FFFFh-0009h	<Value> *16 Ops.

MSRC001_1034 [IBS Op Logical Address] (Core::X86::Msr::IBS_OP_RIP)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1034

Bits	Description
63:0	**IbsOpRip**: **micro-op linear address**. Read-write,Volatile. Reset: 0000_0000_0000_0000h. Linear address in canonical form for the instruction that contains the tagged micro-op.

MSRC001_1035 [IBS Op Data] (Core::X86::Msr::IBS_OP_DATA)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1035

Bits	Description
63:41	Reserved.
40	**IbsOpMicrocode**. Read-write,Volatile. Reset: 0. 1=Tagged operation from microcode.
39	**IbsOpBrnFuse**: **fused branch micro-op**. Read-write,Volatile. Reset: 0. 1=Tagged operation was a fused branch micro-op. Support indicated by Core::X86::Cpuid::IbsIdEax[OpBrnFuse].
38	**IbsRipInvalid**: **RIP is invalid**. Read-write,Volatile. Reset: 0. 1=Tagged operation RIP is invalid. Support indicated by Core::X86::Cpuid::IbsIdEax[RipInvalidChk].
37	**IbsOpBrnRet**: **branch micro-op retired**. Read-write,Volatile. Reset: 0. 1=Tagged operation was a branch micro-op that retired.
36	**IbsOpBrnMisp**: **mispredicted branch micro-op**. Read-write,Volatile. Reset: 0. 1=Tagged operation was a

	branch micro-op that was mispredicted. Qualified by IbsOpBrnRet == 1.
35	**IbsOpBrnTaken**: **taken branch micro-op**. Read-write,Volatile. Reset: 0. 1=Tagged operation was a branch micro-op that was taken. Qualified by IbsOpBrnRet == 1.
34	**IbsOpReturn**: **return micro-op**. Read-write,Volatile. Reset: 0. 1=Tagged operation was return micro-op. Qualified by (IbsOpBrnRet == 1).
33:32	Reserved.
31:16	**IbsTagToRetCtr**: **micro-op tag to retire count**. Read-write,Volatile. Reset: 0000h. This field returns the number of cycles from when the micro-op was tagged to when the micro-op was retired. This field is equal to IbsCompToRetCtr when the tagged micro-op is a NOP.
15:0	**IbsCompToRetCtr**: **micro-op completion to retire count**. Read-write,Volatile. Reset: 0000h. This field returns the number of cycles from when the micro-op was completed to when the micro-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 operation crosses a cache line boundary, the data returned in this register is the data for the access to the lower cache line.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1036

Bits	Description
63:6	Reserved.
5	**CacheHitSt**: **IBS cache hit state**. Read-write,Volatile. Reset: 0. 0=M State. 1=O State. Valid when the data source type is Cache(2h).
4	**RmtNode**: **IBS request destination node**. Read-write,Volatile. Reset: 0. 0=The request is serviced by the NB in the same node as the core. 1=The request is serviced by the NB in a different node than the core. Valid when NbIbsReqSrc is non-zero.
3	Reserved.
2:0	**DataSrc**: **northbridge IBS request data source**. Read-write. Reset: 0h.

ValidValues:

Value	Description
0h	No valid status.
1h	Reserved.
2h	Cache: data returned from another cores cache.
3h	DRAM: data returned from DRAM.
4h	Reserved for remote cache.
6h-5h	Reserved.
7h	Other: data returned from MMIO/Config/PCI/APIC.

MSRC001_1037 [IBS Op Data 3] (Core::X86::Msr::IBS_OP_DATA3)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

If a load or store operation crosses a 256-bit boundary, the data returned in this register is the data for the access to the data below the 256-bit boundary.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1037

Bits	Description
63:48	**IbsTlbRefillLat**: **L1 DTLB refill latency**. 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.
47:32	**IbsDcMissLat**: **data cache miss latency**. 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 data cache writes or prefetch instructions.
31:26	**IbsOpDcMissOpenMemReqs**: **outstanding memory requests on DC fill**. 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	**IbsOpMemWidth**: **load/store size in bytes**. Read-write,Volatile. Reset: 0h. Report the number of bytes the load

<table>
<tr><td></td><td>or store is attempting to access.
<b>ValidValues</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.</td></tr>
<tr><td>4h</td><td>QW.</td></tr>
<tr><td>5h</td><td>OW.</td></tr>
<tr><td>Fh-6h</td><td>Reserved.</td></tr>
</table></td></tr>
<tr><td>21</td><td><b>IbsSwPf</b>: <b>software prefetch</b>. Read-write,Volatile. Reset: 0. 1=The op is a software prefetch.</td></tr>
<tr><td>20</td><td><b>IbsL2Miss</b>: <b>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.</td></tr>
<tr><td>19</td><td><b>IbsDcL2TlbHit1G</b>: <b>data cache L2TLB 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 L2TLB.</td></tr>
<tr><td>18</td><td><b>IbsDcPhyAddrValid</b>: <b>data cache physical address valid</b>. Read-write,Volatile. Reset: 0. 1=The physical address in Core::X86::Msr::IBS_DC_PHYSADDR is valid for the load or store operation.</td></tr>
<tr><td>17</td><td><b>IbsDcLinAddrValid</b>: <b>data cache linear address valid</b>. Read-write,Volatile. Reset: 0. 1=The linear address in Core::X86::Msr::IBS_DC_LINADDR is valid for the load or store operation.</td></tr>
<tr><td>16</td><td><b>DcMissNoMabAlloc</b>: <b>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.</td></tr>
<tr><td>15</td><td><b>IbsDcLockedOp</b>: <b>locked operation</b>. Read-write,Volatile. Reset: 0. 1=Tagged load or store operation is a locked operation.</td></tr>
<tr><td>14</td><td><b>IbsDcUcMemAcc</b>: <b>UC memory access</b>. Read-write,Volatile. Reset: 0. 1=Tagged load or store operation accessed uncacheable memory.</td></tr>
<tr><td>13</td><td><b>IbsDcWcMemAcc</b>: <b>WC memory access</b>. Read-write,Volatile. Reset: 0. 1=Tagged load or store operation accessed write combining memory.</td></tr>
<tr><td>12:9</td><td>Reserved.</td></tr>
<tr><td>8</td><td><b>IbsDcMisAcc</b>: <b>misaligned access</b>. Read-write,Volatile. Reset: 0. 1=The tagged load or store operation crosses a 256 bit address boundary.</td></tr>
<tr><td>7</td><td><b>IbsDcMiss</b>: <b>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.</td></tr>
<tr><td>6</td><td><b>IbsDcL2tlbHit2M</b>: <b>data cache L2TLB 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 L2TLB.</td></tr>
<tr><td>5</td><td><b>IbsDcL1TlbHit1G</b>: <b>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.</td></tr>
<tr><td>4</td><td><b>IbsDcL1TlbHit2M</b>: <b>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.</td></tr>
<tr><td>3</td><td><b>IbsDcL2TlbMiss</b>: <b>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.</td></tr>
<tr><td>2</td><td><b>IbsDcL1tlbMiss</b>: <b>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.</td></tr>
<tr><td>1</td><td><b>IbsStOp</b>: <b>store op</b>. Read-write,Volatile. Reset: 0. 1=Tagged operation is a store operation.</td></tr>
<tr><td>0</td><td><b>IbsLdOp</b>: <b>load op</b>. Read-write,Volatile. Reset: 0. 1=Tagged operation is a load operation.</td></tr>
</table>

MSRC001_1038 [IBS DC Linear Address] (Core::X86::Msr::IBS_DC_LINADDR)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1038

Bits	Description
63:0	**IbsDcLinAd**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_1039

Bits	Description
63:48	Reserved.
47:0	**IbsDcPhysAd**: **load or store physical address**. 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. This field contains valid data only if Core::X86::Msr::IBS_OP_DATA3[IbsDcPhyAddrValid] is asserted.

MSRC001_103A [IBS Control] (Core::X86::Msr::IBS_CTL)

Read,Error-on-write.

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_103A

Bits	Description
63:9	Reserved.
8	**LvtOffsetVal**: **local vector table offset valid**. Read,Error-on-write. Reset: X.
7:4	Reserved.
3:0	**LvtOffset**: **local vector table offset**. Read,Error-on-write. Reset: Xh.

MSRC001_103B [IBS Branch Target Address] (Core::X86::Msr::BP_IBSTGT_RIP)

Read-write,Volatile. Reset: 0000_0000_0000_0000h.

Support for this register indicated by Core::X86::Cpuid::IbsIdEax[BrnTrgt].

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_103B

Bits	Description
63:0	**IbsBrTarget**. Read-write,Volatile. Reset: 0000_0000_0000_0000h. The logical address in canonical form for the branch target. Contains a valid target if non-0. Qualified by Core::X86::Msr::IBS_OP_DATA[IbsOpBrnRet] == 1.

MSRC001_103C [IBS Fetch Control Extended] (Core::X86::Msr::IC_IBS_EXTD_CTL)

Read-only,Volatile. Reset: 0000_0000_0000_0000h.

Support for this register indicated by Core::X86::Cpuid::IbsIdEax[IbsFetchCtlExtd].

_lthree[1:0]_core[3:0]_thread[1:0]; MSRC001_103C

Bits	Description
63:16	Reserved.
15:0	**IbsItlbRefillLat**: **ITLB Refill Latency for the sampled fetch, if there is a reload**. Read-only,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.

2.1.15 Performance Monitor Counters

Any undefined UnitMask bits should be set to zero when selecting an event.

2.1.15.1 RDPMC Assignments

There are six core performance event counters per thread, six performance events counters per L3 complex and four Data Fabric performance events counters mapped to the RDPMC instruction as follows:

- The RDPMC[5:0] instruction accesses core events. See 2.1.15.4 [Core Performance Monitor Counters].
- The RDPMC[9:6] instruction accesses data fabric events.
- The RDPMC[F:A] instruction accesses L3 cache events. See 2.1.15.5 [L3 Cache Performance Monitor Counters].

2.1.15.2 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 19 [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_CTL.
  - 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.

Table 19: Guidance for Common Performance Statistics with Complex Event Selects

Description	Equation
Execution-Time Branch Misprediction Ratio (Non-Speculative)	Event[0x4300C3] / Event[0x4300C2]
L1 Data Cache Accesses	Event[0x430729]
All L2 Cache Accesses	Event[0x43F960] + Event[0x433F70] + Event[0x433F71] + Event[0x433F72]
L2 Cache Accesses: Instruction Cache Misses	Event[0x431060] + Event[0x431861]
L2 Cache Accesses: Data Cache Misses	Event[0x43C860]
L2 Cache Accesses: L2 Prefetcher	Event[0x433F70] + Event[0x433F71] + Event[0x433F72]
L2 Cache Miss	Event[0x430964] + Event[0x433F71] + Event[0x433F72]
L2 Cache Miss from Instruction Cache Miss	Event[0x430164]
L2 Cache Miss from Data Cache Miss	Event[0x430864]
L2 Cache Miss from Prefetch to L2 Cache	Event[0x433F71] + Event[0x433F72]
L2 Cache Hit	Event[0x43F664] + Event[0x433F70]
L2 Cache Hit from IC Miss	Event[0x430664]
L2 Cache Hit from DC Miss	Event[0x437064]
L2 Cache Hit from Prefetch to L2 Cache	Event[0x433F70]
L3 Cache Accesses	L3Event[0xFF0F0000_0040FF04]
L3 Cache Miss	L3Event[0xFF0F0000_00400106]
Average L3 Read Miss Latency (in core clocks)	L3Event[0xFF0F0000_00400090] * 16 / L3Event[0xFF0F0000_00401B9A]
L1 Instruction TLB Misses	Event[0x430084] + Event[0x430785]
L2 Instruction TLB Misses & Instruction page walk	Event[0x430785]
L1 Data TLB Misses	Event[0x43FF45]
L2 Data TLB Misses & Data page walk	Event[0x43F045]
TLBs Flushed	Event[0x43DF78]
Micro-ops Dispatched	Event[0x4303AA]
Micro-ops Retired	Event[0x4300C1]
Outbound data bytes transferred across all remote links on a local node.	(DFEvent[0x000100400287] + DFEvent[0x0001004002C7] + DFEvent[0x000200400207] + DFEvent[0x000200400287] + DFEvent[0x000200400247] + DFEvent[0x0002004002C7]) * 16B
Data bytes transferred across all DRAM channels of a local Node (approximate).	(DFEvent[0x0403807] + DFEvent[0x0403847])*64B

2.1.15.3 Large Increment per Cycle Events

The maximum increment for a regular performance event is 15 (i.e., a 4-bit event). However some event types can have a large increment every cycle (example: Core::X86::Pmc::Core::FpRetSseAvxOps).

An option is provided for merging a pair of even/odd performance monitors to acquire an accurate count. The even performance monitor is programmed with the desired event (example: Core::X86::Pmc::Core::FpRetSseAvxOps). First the odd numbered Core::X86::Msr::PERF_CTL is programmed with the event Core::X86::Pmc::Core::Merge (PMCxFFF) with the enable bit (En) turned on and with the remaining bits off. Then the corresponding even numbered Core::X86::Msr::PERF_CTL is programmed with the desired PMC event. 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., less than 5-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.

PMCxFFF [Merge] (Core::X86::Pmc::Core::Merge)

See 2.1.15.3 [Large Increment per Cycle Events].

PMCxFFF

Bits	Description
7:0	Reserved.

2.1.15.4 Core Performance Monitor Counters

This section provides the core performance counter events that may be selected through Core::X86::Msr::PERF_CTL[EventSelect[11:8],EventSelect[7:0],UnitMask]. See Core::X86::Msr::PERF_CTR. See Core::X86::Msr::PERF_LEGACY_CTL and Core::X86::Msr::PERF_LEGACY_CTR.

2.1.15.4.1 Floating Point (FP) Events

PMCx000 [FPU Pipe Assignment] (Core::X86::Pmc::Core::FpuPipeAssignment)

Read-only. Reset: 00h.

The number of operations (uOps) and dual-pipeuOps dispatched to each of the 4 FPU execution pipelines. This event reflects how busy the FPU pipelines are and may be used for workload characterization. This includes all operations performed by x87, MMX, and SSE instructions, including moves. Each increment represents a one-cycle dispatch event. This event is a speculative event. (See Core::X86::Pmc::Core::ExRetMmxFpInstr). Since this event includes non-numeric operations it is not suitable for measuring MFLOPs.

PMCx000

Bits	Description
7	**Dual3**: **Total number multi-pipe uOps assigned to Pipe 3**. Read-only. Reset: 0.
6	**Dual2**: **Total number multi-pipe uOps assigned to Pipe 2**. Read-only. Reset: 0.
5	**Dual1**: **Total number multi-pipe uOps assigned to Pipe 1**. Read-only. Reset: 0.
4	**Dual0**: **Total number multi-pipe uOps assigned to Pipe 0**. Read-only. Reset: 0.
3	**Total3**: **Total number uOps assigned to Pipe 3**. Read-only. Reset: 0.
2	**Total2**: **Total number uOps assigned to Pipe 2**. Read-only. Reset: 0.

1	**Total1**: **Total number uOps assigned to Pipe 1**. Read-only. Reset: 0.
0	**Total0**: **Total number uOps assigned to Pipe 0**. Read-only. Reset: 0.

PMCx001 [FP Scheduler Empty] (Core::X86::Pmc::Core::FpSchedEmpty)

This is a speculative event. The number of cycles in which the FPU scheduler is empty. Note that some Ops like FP loads bypass the scheduler. Invert this (Core::X86::Msr::PERF_CTL[Inv] == 1) to count cycles in which at least one FPU operation is present in the FPU.

PMCx001

Bits	Description
7:0	Reserved.

PMCx002 [Retired x87 Floating Point Operations] (Core::X86::Pmc::Core::FpRetx87FpOps)

Read-write. Reset: 00h.

The number of x87 floating-point Ops that have retired. The number of events logged per cycle can vary from 0 to 8.

PMCx002

Bits	Description
7:3	Reserved.
2	**DivSqrROps**: **Divide and square root Ops**. Read-write. Reset: 0.
1	**MulOps**: **Multiply Ops**. Read-write. Reset: 0.
0	**AddSubOps**: **Add/subtract Ops**. Read-write. Reset: 0.

PMCx003 [Retired SSE/AVX Operations] (Core::X86::Pmc::Core::FpRetSseAvxOps)

Read-write. Reset: 00h.

This is a retire-based event. The number of retired SSE/AVX FLOPs. The number of events logged per cycle can vary from 0 to 64. This event can count above 15. See 2.1.15.3 [Large Increment per Cycle Events].

PMCx003

Bits	Description
7	**DpMultAddFLOPs**: **Double precision multiply-add FLOPs**. Read-write. Reset: 0. Multiply-add counts as 2 FLOPs.
6	**DpDivFLOPs**: **Double precision divide/square root FLOPs**. Read-write. Reset: 0.
5	**DpMultFLOPs**: **Double precision multiply FLOPs**. Read-write. Reset: 0.
4	**DpAddSubFLOPs**: **Double precision add/subtract FLOPs**. Read-write. Reset: 0.
3	**SpMultAddFLOPs**: **Single precision multiply-add FLOPs**. Read-write. Reset: 0. Multiply-add counts as 2 FLOPs.
2	**SpDivFLOPs**: **Single-precision divide/square root FLOPs**. Read-write. Reset: 0.
1	**SpMultFLOPs**: **Single-precision multiply FLOPs**. Read-write. Reset: 0.
0	**SpAddSubFLOPs**: **Single-precision add/subtract FLOPs**. Read-write. Reset: 0.

PMCx004 [Number of Move Elimination and Scalar Op Optimization] (Core::X86::Pmc::Core::FpNumMovElimScalOp)

Read-write. Reset: 00h.

This is a dispatch based speculative event, and is useful for measuring the effectiveness of the Move elimination and Scalar code optimization schemes.

PMCx004

Bits	Description
7:4	Reserved.
3	**Optimized**: **Number of Scalar Ops optimized**. Read-write. Reset: 0.
2	**OptPotential**: **Number of Ops that are candidates for optimization (have Z-bit either set or pass)**. Read-write. Reset: 0.
1	**SseMovOpsElim**: **Number of SSE Move Ops eliminated**. Read-write. Reset: 0.
0	**SseMovOps**: **Number of SSE Move Ops**. Read-write. Reset: 0.

PMCx005 [Retired Serializing Ops] (Core::X86::Pmc::Core::FpRetiredSerOps)

Read-write. Reset: 00h.

The number of serializing Ops retired.

PMCx005

Bits	Description
7:4	Reserved.
3	**X87CtrlRet**: **x87 control word mispredict traps due to mispredictions in RC or PC, or changes in mask bits**. Read-write. Reset: 0.
2	**X87BotRet**: **x87 bottom-executing uOps retired**. Read-write. Reset: 0.
1	**SseCtrlRet**: **SSE control word mispredict traps due to mispredictions in RC, FTZ or DAZ, or changes in mask bits**. Read-write. Reset: 0.
0	**SseBotRet**: **SSE bottom-executing uOps retired**. Read-write. Reset: 0.

2.1.15.4.2 LS Events

PMCx025 [Retired Lock Instructions] (Core::X86::Pmc::Core::LsLocks)

Reset: 00h.

PMCx025

Bits	Description
7:0	Reserved.

PMCx026 [Retired CLFLUSH Instructions] (Core::X86::Pmc::Core::LsRetClClush)

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::LsRetCpuid)

The number of CPUID instructions retired.

PMCx027

Bits	Description
7:0	Reserved.

PMCx029 [LS Dispatch] (Core::X86::Pmc::Core::LsDispatch)

Read-write. Reset: 00h.

Counts the number of operations dispatched to the LS unit. Unit Masks ADDed.

PMCx029

Bits	Description
7:3	Reserved.
2	**LdStDispatch**: **Load-op-Stores**. Read-write. Reset: 0.
1	**StoreDispatch**. Read-write. Reset: 0.
0	**LdDispatch**. Read-write. Reset: 0.

PMCx02B [SMIs Received] (Core::X86::Pmc::Core::LsSmiRx)

Counts the number of SMIs received.

PMCx02B

Bits	Description
7:0	Reserved.

PMCx02C [Interrupts Taken] (Core::X86::Pmc::Core::LsIntTaken)

Counts the number of interrupts taken.

PMCx02C

Bits	Description
7:0	Reserved.

PMCx035 [Store to Load Forward] (Core::X86::Pmc::Core::LsSTLF)

Number of STLF hits.

PMCx035

Bits	Description
7:0	Reserved.

PMCx037 [Store Commit Cancels 2] (Core::X86::Pmc::Core::LsStCommitCancel2)

Read-write. Reset: 00h.

PMCx037

Bits	Description
7:1	Reserved.
0	**StCommitCancelWcbFull**. Read-write. Reset: 0. A non-cacheable store and the non-cacheable commit buffer is full.

PMCx040 [Data Cache Accesses] (Core::X86::Pmc::Core::LsDcAccesses)

The number of accesses to the data cache for load and store references. This may include certain microcode scratchpad accesses, although these are generally rare. Each increment represents an eight-byte access, although the instruction may only be accessing a portion of that. This event is a speculative event.

PMCx040

Bits	Description
7:0	Reserved.

PMCx041 [DC Miss By Type] (Core::X86::Pmc::Core::LsMabAlloc)

Read-write. Reset: 00h.

PMCx041

Bits	Description
7:4	Reserved.
3	**DcPrefetcher**. Read-write. Reset: 0.
2	Reserved.
1	**Stores**. Read-write. Reset: 0.
0	**Loads**. Read-write. Reset: 0.

PMCx043 [Data Cache Refills from System] (Core::X86::Pmc::Core::LsRefillsFromSys)

Read-write. Reset: 00h.

Demand Data Cache Fills by Data Source.

PMCx043

Bits	Description
7	Reserved.
6	**LS_MABRESP_RMT_DRAM**. Read-write. Reset: 0. DRAM or IO from different die.
5	Reserved.
4	**LS_MABRESP_RMT_CACHE**. Read-write. Reset: 0. Hit in cache; Remote CCX and the address's Home Node is on a different die.
3	**LS_MABRESP_LCL_DRAM**. Read-write. Reset: 0. DRAM or IO from this thread's die.
2	Reserved.
1	**LS_MABRESP_LCL_CACHE**. Read-write. Reset: 0. Hit in cache; local CCX (not Local L2), or Remote CCX and the address's Home Node is on this thread's die.
0	**MABRESP_LCL_L2**. Read-write. Reset: 0. Local L2 hit.

PMCx045 [L1 DTLB Miss] (Core::X86::Pmc::Core::LsL1DTlbMiss)

Read-write. Reset: 00h.

PMCx045

Bits	Description
7	**TlbReload1GL2Miss**. Read-write. Reset: 0.
6	**TlbReload2ML2Miss**. Read-write. Reset: 0.
5	**TlbReload32KL2Miss**. Read-write. Reset: 0.
4	**TlbReload4KL2Miss**. Read-write. Reset: 0.
3	**TlbReload1GL2Hit**. Read-write. Reset: 0.
2	**TlbReload2ML2Hit**. Read-write. Reset: 0.
1	**TlbReload32KL2Hit**. Read-write. Reset: 0.
0	**TlbReload4KL2Hit**. Read-write. Reset: 0.

PMCx046 [Total Page Table Walks] (Core::X86::Pmc::Core::LsTablewalker)

Read-write. Reset: 00h.

PMCx046

Bits	Description
7:4	Reserved.
3	**IcType1**. Read-write. Reset: 0.
2	**IcType0**. Read-write. Reset: 0.
1	**DcType1**. Read-write. Reset: 0.
0	**DcType0**. Read-write. Reset: 0.

PMCx047 [Misaligned loads] (Core::X86::Pmc::Core::LsMisalAccesses)

PMCx047

Bits	Description
7:0	Reserved.

PMCx04B [Prefetch Instructions Dispatched] (Core::X86::Pmc::Core::LsPrefInstrDisp)

Reset: 00h.

Software Prefetch Instructions Dispatched (Speculative).

PMCx04B

Bits	Description
7:0	Reserved.

PMCx052 [Ineffective Software Prefetchs] (Core::X86::Pmc::Core::LsInefSwPref)

Read-write. Reset: 00h.

The number of software prefetches that did not fetch data outside of the processor core.

PMCx052

Bits	Description
7:2	Reserved.
1	**MabMchCnt**. Read-write. Reset: 0. Software PREFETCH instruction saw a match on an already-allocated miss request buffer.
0	**DataPipeSwPfDcHit**. Read-write. Reset: 0. Software PREFETCH instruction saw a DC hit.

PMCx059 [Software Prefetch Data Cache Fills] (Core::X86::Pmc::Core::LsSwPfDcFills)

Read-write. Reset: 00h.

Software Prefetch Data Cache Fills by Data Source.

PMCx059

Bits	Description
7	Reserved.
6	**LS_MABRESP_RMT_DRAM**. Read-write. Reset: 0. DRAM or IO from different die.

5	Reserved.
4	**LS_MABRESP_RMT_CACHE**. Read-write. Reset: 0. Hit in cache; Remote CCX and the address's Home Node is on a different die.
3	**LS_MABRESP_LCL_DRAM**. Read-write. Reset: 0. DRAM or IO from this thread's die.
2	Reserved.
1	**LS_MABRESP_LCL_CACHE**. Read-write. Reset: 0. Hit in cache; local CCX (not Local L2), or Remote CCX and the address's Home Node is on this thread's die.
0	**MABRESP_LCL_L2**. Read-write. Reset: 0. Local L2 hit.

PMCx05A [Hardware Prefetch Data Cache Fills] (Core::X86::Pmc::Core::LsHwPfDcFills)

Read-write. Reset: 00h.

Hardware Prefetch Data Cache Fills by Data Source.

PMCx05A

Bits	Description
7	Reserved.
6	**LS_MABRESP_RMT_DRAM**. Read-write. Reset: 0. DRAM or IO from different die.
5	Reserved.
4	**LS_MABRESP_RMT_CACHE**. Read-write. Reset: 0. Hit in cache; Remote CCX and the address's Home Node is on a different die.
3	**LS_MABRESP_LCL_DRAM**. Read-write. Reset: 0. DRAM or IO from this thread's die.
2	Reserved.
1	**LS_MABRESP_LCL_CACHE**. Read-write. Reset: 0. Hit in cache; local CCX (not Local L2), or Remote CCX and the address's Home Node is on this thread's die.
0	**MABRESP_LCL_L2**. Read-write. Reset: 0. Local L2 hit.

PMCx05B [Table Walker Data Cache Fills by Data Source] (Core::X86::Pmc::Core::LsTwDcFills)

Read-write. Reset: 00h.

PMCx05B

Bits	Description
7	Reserved.
6	**LS_MABRESP_RMT_DRAM**. Read-write. Reset: 0. DRAM or IO from different die.
5	Reserved.
4	**LS_MABRESP_RMT_CACHE**. Read-write. Reset: 0. Hit in cache; Remote CCX and the address's Home Node is on a different die.
3	**LS_MABRESP_LCL_DRAM**. Read-write. Reset: 0. DRAM or IO from this thread's die.
2	Reserved.
1	**LS_MABRESP_LCL_CACHE**. Read-write. Reset: 0. Hit in cache; local CCX (not Local L2), or Remote CCX and the address's Home Node is on this thread's die.
0	**MABRESP_LCL_L2**. Read-write. Reset: 0. Local L2 hit.

PMCx076 [Cycles not in Halt] (Core::X86::Pmc::Core::LsNotHaltedCyc)

PMCx076

Bits	Description
7:0	Reserved.

PMCx078 [All TLB Flushes] (Core::X86::Pmc::Core::LsTlbFlush)

Reset: 00h.

PMCx078

Bits	Description
7:0	Reserved.

2.1.15.4.3 IC and BP Events

Note: All instruction cache events are speculative events unless specified otherwise.

PMCx080 [Total Number of 32B Instruction Fetches] (Core::X86::Pmc::Core::IcFw32)

The number of 32B instruction cache fetches issued to the instruction decoder.

PMCx080

Bits	Description
7:0	Reserved.

PMCx081 [Number of 32B Instruction Fetches that Miss in Instruction Cache] (Core::X86::Pmc::Core::IcFw32Miss)

The number of 32B fetches that tried to read the L1 instruction cache and missed.

PMCx081

Bits	Description
7:0	Reserved.

PMCx082 [Instruction Cache Refills from L2] (Core::X86::Pmc::Core::IcCacheFillL2)

The number of 64 byte instruction cache line was fulfilled from the L2 cache.

PMCx082

Bits	Description
7:0	Reserved.

PMCx083 [Instruction Cache Refills from System] (Core::X86::Pmc::Core::IcCacheFillSys)

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, L2 ITLB Hit] (Core::X86::Pmc::Core::BpL1TlbMissL2TlbHit)

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::BpL1TlbMissL2TlbMiss)

The number of instruction fetches that miss in both the L1 and L2 TLBs.

PMCx085

Bits	Description
7:0	Reserved.

PMCx08A [L1 Branch Prediction Overrides Existing Prediction (speculative)] (Core::X86::Pmc::Core::BpL1BTBCorrect)

PMCx08A

Bits	Description
7:0	Reserved.

PMCx08B [L2 Branch Prediction Overrides Existing Prediction (speculative)] (Core::X86::Pmc::Core::BpL2BTBCorrect)

PMCx08B

Bits	Description
7:0	Reserved.

PMCx08C [Instruction Cache Lines Invalidated] (Core::X86::Pmc::Core::IcCacheInval)

Read-write. Reset: 00h.

The number of instruction cache lines invalidated. A non-SMC event is CMC (cross modifying code), either from the other thread of the core or another core.

PMCx08C

Bits	Description
7:2	Reserved.
1	**L2InvalidatingProbe**. Read-write. Reset: 0. IC line invalidated due to L2 invalidating probe (external or LS).
0	**FillInvalidated**. Read-write. Reset: 0. IC line invalidated due to overwriting fill response.

PMCx08E [Dynamic Indirect Predictions] (Core::X86::Pmc::Core::BpDynIndPred)

Indirect Branch Prediction for potential multi-target branch (speculative)

PMCx08E

Bits	Description
7:0	Reserved.

PMCx091 [Decoder Overrides Existing Branch Prediction (speculative)] (Core::X86::Pmc::Core::BpDeReDirect)

PMCx091

Bits	Description
7:0	Reserved.

PMCx099 [ITLB Reloads] (Core::X86::Pmc::Core::BpTlbRel)

The number of ITLB reload requests.

PMCx099

Bits	Description
7:0	Reserved.

PMCx28A [Switches between instruction fetch and op cache fetch modes] (Core::X86::Pmc::Core::IcOcModeSwitch)

Read-write. Reset: 00h.

See Core::X86::Pmc::Core::DeDisUopsFromDecoder (PMCx0AA) for number of ops dispatched from OC.

PMCx28A

Bits	Description
7:2	Reserved.
1	**OcIcModeSwitch**: **OC to IC mode switch**. Read-write. Reset: 0.
0	**IcOcModeSwitch**: **IC to OC mode switch**. Read-write. Reset: 0.

2.1.15.4.4 DE Events

PMCx0AA [UOps Dispatched From Decoder] (Core::X86::Pmc::Core::DeDisUopsFromDecoder)

Read-write. Reset: 00h.

Ops dispatched from either the decoders, OpCache or both.

PMCx0AA

Bits	Description
7:2	Reserved.
1	**OpCacheDispatched**: **Count of dispatched Ops from OpCache**. Read-write. Reset: 0.
0	**DecoderDispatched**: **Count of dispatched Ops from Decoder**. Read-write. Reset: 0.

PMCx0AE [Dispatch Resource Stall Cycles 1] (Core::X86::Pmc::Core::DeDisDispatchTokenStalls1)

Read-write. Reset: 00h.

Cycles where a dispatch group is valid but does not get dispatched due to a Token Stall.

PMCx0AE

Bits	Description
7	**FPMiscRsrcStall**: **FP Miscellaneous resource unavailable**. Read-write. Reset: 0. Applies to the recovery of mispredicts with FP ops.
6	**FPSchRsrcStall**: **FP scheduler resource stall**. Read-write. Reset: 0. Applies to ops that use the FP scheduler.
5	**FpRegFileRsrcStall**: **floating point register file resource stall**. Read-write. Reset: 0. Applies to all FP ops that have a destination register.
4	**TakenBrnchBufferRsrc**: **taken branch buffer resource stall**. Read-write. Reset: 0.
3	**IntSchedulerMiscRsrcStall**: **Integer Scheduler miscellaneous resource stall**. Read-write. Reset: 0.
2	**StoreQueueRsrcStall**: **Store Queue resource stall**. Read-write. Reset: 0. Applies to all ops with store semantics.
1	**LoadQueueRsrcStall**: **Load Queue resource stall**. Read-write. Reset: 0. Applies to all ops with load semantics.
0	**IntPhyRegFileRsrcStall**: **Integer Physical Register File resource stall**. Read-write. Reset: 0. Integer Physical Register File, applies to all ops that have an integer destination register.

PMCx0AF [**Dispatch Resource Stall Cycles 0**] **(Core::X86::Pmc::Core::DeDisDispatchTokenStalls0)**

Read-write. Reset: 00h.

Cycles where a dispatch group is valid but does not get dispatched due to a token stall.

PMCx0AF

Bits	Description
7	Reserved.
6	**RetireRsrcStall**: **RETIRE Resource unavailable**. Read-write. Reset: 0.
5	**AGSQRsrcStall**: **AGSQ Resources unavailable**. Read-write. Reset: 0.
4	**ALURsrcStall**: **ALU Resources total unavailable**. Read-write. Reset: 0.
3	**ALSQ3_0_RsrcStall**. Read-write. Reset: 0.
2	**ALSQ3RsrcStall**: **ALSQ 3 Resources unavailable**. Read-write. Reset: 0.
1	**ALSQ2RsrcStall**: **ALSQ 2 Resources unavailable**. Read-write. Reset: 0.
0	**ALSQ1RsrcStall**: **ALSQ 1 Resources unavailable**. Read-write. Reset: 0.

2.1.15.4.5 EX (SC) Events

PMCx0C0 [**Retired Instructions**] **(Core::X86::Pmc::Core::ExRetInstr)**

PMCx0C0

Bits	Description
7:0	Reserved.

PMCx0C1 [**Retired Uops**] **(Core::X86::Pmc::Core::ExRetCops)**

The number of micro-ops retired. This count includes all processor activity (instructions, exceptions, interrupts, microcode assists, etc.). The number of events logged per cycle can vary from 0 to 8.

PMCx0C1

Bits	Description
7:0	Reserved.

PMCx0C2 [**Retired Branch Instructions**] **(Core::X86::Pmc::Core::ExRetBrn)**

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::ExRetBrnMisp)**

The number of branch instructions retired, of any type, that were not correctly predicted. This includes those for which prediction is not attempted (far control transfers, exceptions and interrupts).

PMCx0C3

Bits	Description
7:0	Reserved.

PMCx0C4 [Retired Taken Branch Instructions] (Core::X86::Pmc::Core::ExRetBrnTkn)

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::ExRetBrnTknMisp)

The number of retired taken branch instructions that were mispredicted.

PMCx0C5

Bits	Description
7:0	Reserved.

PMCx0C6 [Retired Far Control Transfers] (Core::X86::Pmc::Core::ExRetBrnFar)

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 Returns] (Core::X86::Pmc::Core::ExRetNearRet)

The number of near return instructions (RET or RET Iw) retired.

PMCx0C8

Bits	Description
7:0	Reserved.

PMCx0C9 [Retired Near Returns Mispredicted] (Core::X86::Pmc::Core::ExRetNearRetMispred)

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.

PMCx0C9

Bits	Description
7:0	Reserved.

PMCx0CA [Retired Indirect Branch Instructions Mispredicted] (Core::X86::Pmc::Core::ExRetBrnIndMisp)

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::ExRetMmxFpInstr)

Read-write. Reset: 00h.

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	**SseInstr**. Read-write. Reset: 0. SSE instructions (SSE, SSE2, SSE3, SSSE3, SSE4A, SSE41, SSE42, AVX).
1	**MmxInstr**. Read-write. Reset: 0. MMX instructions.
0	**X87Instr**: **x87 instructions**. Read-write. Reset: 0.

PMCx0D1 [Retired Conditional Branch Instructions] (Core::X86::Pmc::Core::ExRetCond)

PMCx0D1

Bits	Description
7:0	Reserved.

PMCx0D3 [Div Cycles Busy count] (Core::X86::Pmc::Core::ExDivBusy)

PMCx0D3

Bits	Description
7:0	Reserved.

PMCx0D4 [Div Op Count] (Core::X86::Pmc::Core::ExDivCount)

PMCx0D4

Bits	Description
7:0	Reserved.

PMCx1CF [Tagged IBS Ops] (Core::X86::Pmc::Core::ExTaggedIbsOps)

Read-write. Reset: 00h.

PMCx1CF

Bits	Description
7:3	Reserved.
2	**IbsCountRollover**. Read-write. Reset: 0. Number of times an op could not be tagged by IBS because of a previous tagged op that has not retired.
1	**IbsTaggedOpsRet**: **Number of Ops tagged by IBS that retired**. Read-write. Reset: 0.
0	**IbsTaggedOps**: **Number of Ops tagged by IBS**. Read-write. Reset: 0.

PMCx1D0 [Retired Fused Branch Instructions] (Core::X86::Pmc::Core::ExRetFusBrnchInst)

The number of fuse-branch instructions retired per cycle. The number of events logged per cycle can vary from 0-8.

PMCx1D0

Bits	Description
7:0	Reserved.

2.1.15.4.6 L2 Cache Events.

PMCx060 [Requests to L2 Group1] (Core::X86::Pmc::Core::L2RequestG1)

Read-write. Reset: 00h.

All L2 Cache Requests (Breakdown 1 - Common).

PMCx060

Bits	Description
7	**RdBlkL**: **Data Cache Reads (including prefetch)**. Read-write. Reset: 0.
6	**RdBlkX**: **Data Cache Stores**. Read-write. Reset: 0.
5	**LsRdBlkC_S**: **Data Cache Shared Reads**. Read-write. Reset: 0.
4	**CacheableIcRead**: **Instruction Cache Reads**. Read-write. Reset: 0.
3	**ChangeToX**: **Data Cache State Change Requests**. Read-write. Reset: 0.
2	**PrefetchL2**: **Software Prefetch**. Read-write. Reset: 0. Assume core should also count these and allow the breakdown between hardware versus software and data versus instruction.
1	**L2HwPf**: **L2 Prefetcher**. Read-write. Reset: 0. All prefetches accepted by L2 pipeline, hit or miss. Types of PF and L2 hit/miss broken out in a separate perfmon event.

0	**Group2**. Read-write. Reset: 0. Miscellaneous events covered in more detail by Core::X86::Pmc::Core::L2RequestG2 (PMCx061).

PMCx061 [Requests to L2 Group2] (Core::X86::Pmc::Core::L2RequestG2)

Read-write. Reset: 00h.

All L2 Cache Requests (Breakdown 2 - Rare).

PMCx061

Bits	Description
7	**Group1**. Read-write. Reset: 0. Miscellaneous events covered in more detail by Core::X86::Pmc::Core::L2RequestG1 (PMCx060).
6	**LsRdSized**: **Data cache read sized**. Read-write. Reset: 0.
5	**LsRdSizedNC**: **Data cache read sized non-cacheable**. Read-write. Reset: 0.
4	**IcRdSized**: **Instruction cache read sized**. Read-write. Reset: 0.
3	**IcRdSizedNC**: **Instruction cache read sized non-cacheable**. Read-write. Reset: 0.
2	**SmcInval**: **Self-modifying core invalidates**. Read-write. Reset: 0.
1	**BusLocksOriginator**: **Bus locks**. Read-write. Reset: 0.
0	**BusLocksResponses**: **Bus Lock Response**. Read-write. Reset: 0.

PMCx062 [L2 Latency] (Core::X86::Pmc::Core::L2Latency)

Read-write. Reset: 00h.

Total cycles spent waiting for L2 fills to complete from L3 or memory, divided by four. This may be used to calculate average latency by multiplying this count by four and then dividing by the total number of L2 fills (unit mask Core::X86::Pmc::Core::L2RequestG1 == FEh). Event counts are for both threads. To calculate average latency, the number of fills from both threads must be used.

PMCx062

Bits	Description
7:1	Reserved.
0	**L2CyclesWaitingOnFills**. Read-write. Reset: 0.

PMCx064 [Core to L2 Cacheable Request Access Status] (Core::X86::Pmc::Core::L2CacheReqStat)

Read-write. Reset: 00h.

L2 Cache Request Outcomes (not including L2 Prefetch).

PMCx064

Bits	Description
7	**LsRdBlkCS**: **Data Cache Shared Read Hit in L2**. Read-write. Reset: 0.
6	**LsRdBlkLHitX**: **Data Cache Read Hit in L2**. Read-write. Reset: 0.
5	**LsRdBlkLHitS**: **Data Cache Read Hit on Shared Line in L2**. Read-write. Reset: 0.
4	**LsRdBlkX**: **Data Cache Store or State Change Hit in L2**. Read-write. Reset: 0.
3	**LsRdBlkC**: **Data Cache Req Miss in L2 (all types)**. Read-write. Reset: 0.
2	**IcFillHitX**: **Instruction Cache Hit Modifiable Line in L2**. Read-write. Reset: 0.
1	**IcFillHitS**: **Instruction Cache Hit Clean Line in L2**. Read-write. Reset: 0.
0	**IcFillMiss**: **Instruction Cache Req Miss in L2**. Read-write. Reset: 0.

PMCx06D [Cycles with fill pending from L2] (Core::X86::Pmc::Core::L2FillPending)

Read-write. Reset: 00h.

Total cycles spent with one or more fill requests in flight from L2.

PMCx06D

Bits	Description
7:1	Reserved.
0	**L2FillBusy**. Read-write. Reset: 0.

PMCx070 [L2 Prefetch Hit in L2] (Core::X86::Pmc::Core::L2PfHitL2)

Reset: 00h.	
PMCx070	
Bits	**Description**
7:0	Reserved.

PMCx071 [**L2 Prefetcher Hits in L3**] **(Core::X86::Pmc::Core::L2PfMissL2HitL2)**

Reset: 00h.	
Counts all L2 prefetches accepted by the L2 pipeline which miss the L2 cache and hit the L3.	
PMCx071	
Bits	**Description**
7:0	Reserved.

PMCx072 [**L2 Prefetcher Misses in L3**] **(Core::X86::Pmc::Core::L2PfMissL2L3)**

Reset: 00h.	
Counts all L2 prefetches accepted by the L2 pipeline which miss the L2 and the L3 caches.	
PMCx072	
Bits	**Description**
7:0	Reserved.

2.1.15.5 L3 Cache Performance Monitor Counters

The L3 cache is organized as 4 2MB slices shared by 4 cores.
This section provides the core performance counter events that may be selected through Core::X86::Msr::ChL3PmcCfg.

- Unless otherwise noted, L3 Perfmon events require the Core::X86::Msr::ChL3PmcCfg[SliceMask] field to be set. If it is not set, the PMC count will be zero.
- Unless otherwise noted, L3 PMC's require Core::X86::Msr::ChL3PmcCfg[ThreadMask] to be set. The events that don't require a ThreadMask will continue to count, even if the ThreadMask is 0. All other events will have a count of 0 when the ThreadMask is 0.
- When in non-SMT mode, ThreadMask must be 0 for events that don't ignore ThreadMask,

2.1.15.5.1 L3 Cache PMC Events

L3PMCx01 [**L3 Cache Accesses**] **(Core::X86::Pmc::L3::L3RequestG1)**

Read-write. Reset: 00h.	
L3PMCx01	
Bits	**Description**
7	**Caching**: **L3 cache accesses**. Read-write. Reset: 0.
6:0	Reserved.

L3PMCx04 [**All L3 Cache Requests**] **(Core::X86::Pmc::L3::L3LookupState)**

Read-write. Reset: 00h.	
L3PMCx04	
Bits	**Description**
7:0	**AllL3ReqTyps**: **All L3 Request Types**. Read-write. Reset: 00h.

L3PMCx06 [**L3 Miss**] **(Core::X86::Pmc::L3::L3CombClstrState)**

Read-write. Reset: 00h.	
L3 Cache Request Outcomes.	
L3PMCx06	
Bits	**Description**

7:1	Reserved.
0	**RequestMiss**: **L3 cache miss**. Read-write. Reset: 0.

L3PMCx90 [L3 Cache Miss Latency] (Core::X86::Pmc::L3::XiSysFillLatency)

Ignores SliceMask and ThreadMask.
Total cycles for all transactions divided by 16.

L3PMCx90

Bits	Description
7:0	Reserved.

L3PMCx9A [L3 Misses by Request Type] (Core::X86::Pmc::L3::XiCcxSdpReq1)

Reset: 00h.

Ignores SliceMask and ThreadMask.

L3PMCx9A

Bits	Description
7:0	Reserved.

2.1.16 Instruction Based Sampling (IBS)

IBS is a code profiling mechanism that enables the processor to select a random instruction fetch or micro-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::Msr::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::Msr::IBS_FETCH_CTL; and instruction execution performance controlled by Core::X86::Msr::IBS_OP_CTL. Instruction fetch sampling provides information about instruction TLB and instruction cache behavior for fetched instructions. Instruction execution sampling provides information about micro-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::Msr::IBS_FETCH_CTL.
- The number of clock cycles spent on the instruction fetch. See Core::X86::Msr::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::Msr::IBS_FETCH_CTL.
- The linear address, physical address associated with the fetch. See Core::X86::Msr::IBS_FETCH_LINADDR, Core::X86::Msr::IBS_FETCH_PHYSADDR.

Instruction execution performance is profiled by tagging one micro-Op associated with an instruction. Instructions that decode to more than one micro-Op return different performance data depending upon which micro-Op associated with the instruction is tagged. These micro-Ops are associated with the RIP of the next instruction to retire. The following performance information is returned for the tagged micro-Op:

- Branch and execution status for micro-ops. See Core::X86::Msr::IBS_OP_DATA.
- Branch target address for branch micro-ops. See Core::X86::Msr::BP_IBSTGT_RIP.
- The logical address associated with the micro-Op. See Core::X86::Msr::IBS_OP_RIP.
- The linear and physical address associated with a load or store micro-Op. See Core::X86::Msr::IBS_DC_LINADDR, Core::X86::Msr::IBS_DC_PHYSADDR.
- The data cache access status associated with the micro-Op: DC hit/miss, DC miss latency, TLB hit/miss, TLB page size. See Core::X86::Msr::IBS_OP_DATA3.
- The number clocks from when the micro-Op was tagged until the micro-Op retires. See

Core::X86::Msr::IBS_OP_DATA.

- The number clocks from when the micro-Op completes execution until the micro-Op retires. See Core::X86::Msr::IBS_OP_DATA.
- Source information for DRAM and MMIO. See Core::X86::Msr::IBS_OP_DATA2.

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. A broad view of the RAS environment can be found in docRAS. This section describes the capabilities provided by the Family 17h Model 00h-0Fh processor.

3.1 Machine Check Architecture

Table 20: Machine Check Terms and Acronyms

Term	Description
MCA	Machine Check Architecture.
MCAX	Machine Check Architecture eXtensions.
WRIG	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 diagnosibility.
    - 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 is portable.

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 is a per-thread value, and reflects the number of MCA banks visible to that thread. 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 thread.
- 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.
- 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 21 [Legacy MCA MSR Layout].

Table 21: 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

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.

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 22 [MCAX MSR Layout].

Access to unused MCAX MSRs is RAZ/WRIG. MCA Bank 4 is always read-as-zero (RAZ/WRIG).

Table 22: MCAX MSR Layout

MCA bank	MCAX MSR (MSRC000_2xxx)										
	Legacy MCA Bank registers				MCAX Bank registers						
	CTL	STATUS	ADDR	MISC0	CONFIG	IPID	SYND	Reserved	DESTATUS	DEADDR	MISC[4:1]
0	000	001	002	003	004	005	006	007	008	009	00D:00A

1	010	011	012	013	014	015	016	017	018	019	01D:01A
2	020	021	022	023	024	025	026	027	028	029	02D:02A
…											
63	3F0	3F1	3F2	3F3	3F4	3F5	3F6	3F7	3F8	3F9	3FD:3FA

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 23 [MCAX Implementation-Specific Register Layout].

Table 23: 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 error 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 24 [Error Overwrite Priorities] indicates which errors are overwritten in the error status registers.

Table 24: Error Overwrite Priorities

		Older Error		
		Uncorrected	Deferred	Corrected
Newer Error	Uncorrected	-	Overwrite	Overwrite
	Deferred	-	-	Overwrite
	Corrected	-	-	-

Table 25 [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 25: 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 26: 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 27: Error code: transaction type (TT)

TT	Transaction Type
00	Instruction
01	Data
10	Generic
11	Reserved

Table 28: Error codes: cache level (LL)

LL	Cache Level
00	
01	L1: Level 1
10	L2: Level 2
11	LG: Generic

Table 29: 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. Counts are 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, 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::Msr::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::Msr::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[Mcax]). An MCA bank is an MCAX bank if MCA_CONFIG[Mcax]=1 in that bank.

3.1.5.2 Communicating MCAX Support

Software which supports MCAX must set MCA_CONFIG[McaxEn]=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::Msr::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::Msr::MCG_CTL register must be programmed identically for all cores in a processor.
- 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 thread.
    - 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.
- 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

Blocks capable of supporting MCA banks are given in Table 30 [Blocks Capable of Supporting MCA Banks]. Whether a particular block implements an MCA bank is processor-specific and can be determined by querying MCA_IPID for each MCA bank. The block type can be uniquely identified by the MCA_IPID[HardwareId] and MCA_IPID[McaType] values in each MCA bank.

Table 30: Blocks Capable of Supporting MCA Banks

Block	Block Function	MCA_IPID[HardwareId]	MCA_IPID[McaType]
LS	Load-Store Unit	0xb0	0x0
IF	Instruction Fetch Unit	0xb0	0x1

L2	L2 Cache Unit	0xb0	0x2
DE	Decode Unit	0xb0	0x3
EX	Execution Unit	0xb0	0x5
FP	Floating Point Unit	0xb0	0x6
L3	L3 Cache Unit	0xb0	0x7
UMC	Unified Memory Controller	0x96	0x0
SMU	System Management Unit	0x01	0x0
PSP	Platform Security Processor	0xFF	0x0
PB	Parameter Block	0x5	0x0
CS	Coherent Slave	0x2E	0x0
PIE	Power Management, Interrupts, Etc.	0x2E	0x1

3.2.2 Implemented Machine Check Bank Registers

Table 31 [Legacy MCA Registers] provides links to the description of each block's Legacy MCA registers. Table 32 [MCAX Registers] provides links to the description of each block's MCA Extension Registers.

Table 31: 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
PB	MCA::PB::MCA_CTL_PB	MCA::PB::MCA_STATUS_PB	MCA::PB::MCA_ADDR_PB	MCA::PB::MCA_MISC0_PB	MCA::PB::MCA_CTL_MASK_PB

Table 32: MCAX Registers

Block	MCA Register				
	CONFIG	IPID	SYND	DESTATUS	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
PB	MCA::PB::MCA_CONFIG_PB	MCA::PB::MCA_IPID_PB	MCA::PB::MCA_SYND_PB	--	--

3.2.3 Mapping of Banks to Blocks

Table 33: MCA Bank to Block Mapping

Bank	Block
0	LS
1	IF
2	L2
3	DE
4	RAZ
5	EX
6	FP
7	L3
8	L3
9	L3
10	L3
11	L3
12	L3
13	L3
14	L3
15	UMC
16	UMC
17	Reserved
18	Reserved
19	PB
20	CS
21	CS
22	PIE

3.2.4 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 30 [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 3.2.5 [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. MCA_DESTAT does not contain an ErrorCodeExt field, so in this case it is not possible to determine the type of error logged in MCA_DESTAT.

3.2.5 MCA Banks

3.2.5.1 LS

MSR0000_0400...MSRC000_2000 [**LS Machine Check Control**] **(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::Msr::MCG_CTL. Does not affect error detection, correction, or logging.

_lthree[1:0]_core[3:0]_inst0_aliasMSRLEGACY; MSR0000_0400

_lthree[1:0]_core[3:0]_inst0_aliasMSR; MSRC000_2000

Bits	Description
63:21	Reserved.
20	**L2DataErr**. Read-write. Reset: 0. L2 Fill Data error.
19	**DcTagErr5**. Read-write. Reset: 0. DC Tag error type 5.
18	**DcTagErr3**. Read-write. Reset: 0. DC Tag error type 3.
17	**PDC**. Read-write. Reset: 0. PDC parity error. MCA_ADDR_LS logs a virtual address.
16	**L2DTLB**. Read-write. Reset: 0. Level 2 TLB parity error. MCA_ADDR_LS logs a virtual address.
15	**DcTagErr4**. Read-write. Reset: 0. DC Tag error type 4.
14	**DcDataErr3**. Read-write. Reset: 0. DC Data error type 3.
13	**DcDataErr2**. Read-write. Reset: 0. DC Data error type 2.
12	**DcDataErr1**. Read-write. Reset: 0. DC Data error type 1 and poison consumption. MCA_STATUS[Poison] is set on poison consumption from L2/L3.
11	**DcTagErr2**. Read-write. Reset: 0. DC Tag error type 2.
10	**SystemReadDataErrorT1**. Read-write. Reset: 0. System Read Data Error Thread 1. An error in a read of a line from the data fabric. Possible reasons include master abort and target abort.
9	**SystemReadDataErrorT0**. Read-write. Reset: 0. System Read Data Error Thread 0. An error in a read of a line from the data fabric. Possible reasons include master abort and target abort.
8	**IntErrTyp2**. Read-write. Reset: 0. Internal error type 2.
7	**IntErrTyp1**. Read-write. Reset: 0. Internal error type 1.
6	**DcTagErr1**. Read-write. Reset: 0. DC Tag error type 1.
5	**DcTagErr6**. Read-write. Reset: 0. DC Tag error type 6.
4	Reserved.
3	**L1DTLB**. Read-write. Reset: 0. Level 1 TLB parity error.
2	**MAB**. Read-write. Reset: 0. Miss address buffer payload parity error.
1	**STQ**. Read-write. Reset: 0. Store queue parity error.
0	**LDQ**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSRLSLEGACY; MSR0000_0001

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSRLEGACY; MSR0000_0401

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSR; MSRC000_2001

Bits	Description
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::LS::MCA_MISC0_LS.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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	**ErrorCode**. 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 34: MCA_STATUS_LS[ErrorCodeExt] Decode

Error Type	**Bits**	**Description**
LDQ	[6:0]	0x0
STQ	[6:0]	0x1
MAB	[6:0]	0x2
L1DTLB	[6:0]	0x3
DcTagErr5	[6:0]	0x4
DcTagErr6	[6:0]	0x5
DcTagErr1	[6:0]	0x6
IntErrTyp1	[6:0]	0x7
IntErrTyp2	[6:0]	0x8
SystemReadDataErrorT0	[6:0]	0x9
SystemReadDataErrorT1	[6:0]	0xa
DcTagErr2	[6:0]	0xb
DcDataErr1	[6:0]	0xc
DcDataErr2	[6:0]	0xd
DcDataErr3	[6:0]	0xe
DcTagErr4	[6:0]	0xf
L2DTLB	[6:0]	0x10
PDC	[6:0]	0x11
DcTagErr3	[6:0]	0x12
DcTagErr5	[6:0]	0x13
L2DataErr	[6:0]	0x14

MSR0000_0000...MSRC000_2002 [LS Machine Check Address Thread 0] (MCA::LS::MCA_ADDR_LS)

Read-write,Volatile. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSRLSLEGACY; MSR0000_0000

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSRLEGACY; MSR0000_0402

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSR; MSRC000_2002

Bits	**Description**
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. 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[55:0] contains a valid byte address. A value of 6 indicates that MCA::LS::MCA_ADDR_LS[55:6] 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[55:12] contain a valid 4KB memory page and that MCA::LS::MCA_ADDR_LS[11:0] should be ignored by error handling software.

55:0	**ErrorAddr**. 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::LS::MCA_STATUS_LS. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 35: MCA_ADDR_LS Register

Error Type	Bits	Description
LDQ	[55:0]	Reserved
STQ	[55:0]	Reserved
MAB	[55:0]	Reserved
L1DTLB	[55:48] [47:12] [11:0]	Reserved Virtual Address Reserved
DcTagErr5	[55:0]	Reserved
DcTagErr6	[55:0]	Reserved
DcTagErr1	[55:0]	Reserved
IntErrTyp1	[55:0]	Reserved
IntErrTyp2	[55:0]	Reserved
SystemReadDataErrorT0	[55:48] [47:6]	Reserved Physical Address
SystemReadDataErrorT1	[55:48] [47:6]	Reserved Physical Address
DcTagErr2	[55:48] [47:6]	Reserved Physical Address
DcDataErr1	[55:48] [47:6]	Reserved Physical Address
DcDataErr2	[55:48] [47:1]	Reserved Physical Address
DcDataErr3	[55:48] [47:1]	Reserved Physical Address
DcTagErr4	[55:0]	Reserved
L2DTLB	[55:48] [47:12] [11:0]	Reserved Virtual Address Reserved
PDC	[55:48] [47:12] [11:0]	Reserved Virtual Address Reserved
DcTagErr3	[55:0]	Reserved
DcTagErr5	[55:0]	Reserved
L2DataErr	[55:48] [47:6]	Reserved Physical Address

MSR0000_0403...MSRC000_2003 [LS Machine Check Miscellaneous 0 Thread 0] (MCA::LS::MCA_MISC0_LS)

Log miscellaneous information associated with errors.

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSRLEGACY; MSR0000_0403

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSR; MSRC000_2003

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.

62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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	**LvtOffset**. 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	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::LS::MCA_MISC0_LS[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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	**Ovrflw**. 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	**ErrCnt**. 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	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2004 [LS Machine Check Configuration] (MCA::LS::MCA_CONFIG_LS)

Reset: 0000_0000_0000_0025h.

Controls configuration of the associated machine check bank.

_lthree[1:0]_core[3:0]_inst0_aliasMSR; MSRC000_2004

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**LogDeferredInMcaStat**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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	**McaX**. 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] (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.

_lthree[1:0]_core[3:0]_inst0_aliasMSR; MSRC000_2005

Bits	Description
63:48	**McaType**. Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. 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,Volatile. Reset: Cold,0000_0000_0000_0000h.

Logs physical location information associated with error in MCA::LS::MCA_STATUS_LS Thread 0

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSR; MSRC000_2006

Bits	Description
63:39	Reserved.
38:32	**Syndrome**. Read-write,Volatile. Reset: Cold,00h. 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	**ErrorPriority**. Read-write,Volatile. 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	**Length**. Read-write,Volatile. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 36 [MCA_SYND_LS Register].

Table 36: MCA_SYND_LS Register

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]	Reserved Index Way
DcTagErr6	[17:16] [15:8] [7:0]	Reserved Index Way
DcTagErr1	[17:16] [15:8] [7:0]	Reserved Index Way
IntErrTyp1	[17] [16] [15:0]	Reserved Thread ID Reserved
IntErrTyp2	[17] [16] [15:0]	Reserved Thread ID Reserved
SystemReadDataErrorT0	[17:2] [1:0]	Reserved 2'b00 = Master Abort; 2'b01 = Target Abort; 2'b10 = Transaction Error; 2'b11 = Protection Violation
SystemReadDataErrorT1	[17:2] [1:0]	Reserved 2'b00 = Master Abort; 2'b01 = Target Abort; 2'b10 = Transaction Error; 2'b11 = Protection Violation
DcTagErr2	[17:16] [15:8] [7:0]	Reserved Index Way
DcDataErr1	[17:16] [15:8] [7:0]	Reserved Index Way
DcDataErr2	[17:16] [15:8] [7:0]	Reserved Index Way
DcDataErr3	[17:16] [15:8] [7:0]	Reserved Index Way
DcTagErr4	[17:16] [15:8] [7:0]	Reserved Index Way
L2DTLB	[17:15] [14:8]	Reserved Reserved

	[7:4] [3:0]	Reserved Reserved
PDC	[17:0]	Reserved
DcTagErr3	[17:16] [15:8] [7:0]	Reserved Index Way
DcTagErr5	[17:16] [15:8] [7:0]	Reserved Index Way
L2DataErr	[17:0]	Reserved

MSRC000_2008 [LS Machine Check Deferred Error Status Thread 0] (MCA::LS::MCA_DESTAT_LS)

Read-write,Volatile. Reset: Cold,0000_0000_0000_0000h.

Holds status information for the first deferred error seen in this bank.

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSR; MSRC000_2008

Bits	Description
63	**Val**. Read-write,Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	**Overflow**. 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	Reserved.
58	**AddrV**. Read-write,Volatile. Reset: Cold,0. 1=MCA::LS::MCA_DEADDR_LS contains address information associated with the error.
57:54	Reserved.
53	**SyndV**. Read-write,Volatile. 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	Reserved.
44	**Deferred**. 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.
43:0	Reserved.

MSRC000_2009 [LS Deferred Error Address Thread 0] (MCA::LS::MCA_DEADDR_LS)

Read-write,Volatile. 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_DEADDR_LS[LSB].

_lthree[1:0]_core[3:0]_thread[1:0]_inst0_aliasMSR; MSRC000_2009

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::LS::MCA_DEADDR_LS[ErrorAddr]. A value of 0 indicates that MCA::LS::MCA_DEADDR_LS[55:0] contains a valid byte address. A value of 6 indicates that MCA::LS::MCA_DEADDR_LS[55:6] contains a valid cache line address and that MCA::LS::MCA_DEADDR_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_DEADDR_LS[55:12] contain a valid 4KB memory page and that MCA::LS::MCA_DEADDR_LS[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. Read-write,Volatile. Reset: Cold,00_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_DEADDR_LS[LSB].

MSRC001_0400 [LS Machine Check Control Mask] (MCA::LS::MCA_CTL_MASK_LS)

Read-write. Reset: 0000_0000_0000_0000h.

Inhibit detection of an error source.

_lthree[1:0]_core[3:0]_inst0_aliasMSR; MSRC001_0400

Bits	Description
63:21	Reserved.
20	**L2DataErr**. Read-write. Reset: 0. L2 Fill Data error.
19	**DcTagErr5**. Read-write. Reset: 0. DC Tag error type 5.
18	**DcTagErr3**. Read-write. Reset: 0. DC Tag error type 3.
17	**PDC**. Read-write. Reset: 0. PDC parity error. MCA_ADDR_LS logs a virtual address.
16	**L2DTLB**. Read-write. Reset: 0. Level 2 TLB parity error. MCA_ADDR_LS logs a virtual address.
15	**DcTagErr4**. Read-write. Reset: 0. DC Tag error type 4.
14	**DcDataErr3**. Read-write. Reset: 0. DC Data error type 3.
13	**DcDataErr2**. Read-write. Reset: 0. DC Data error type 2.
12	**DcDataErr1**. Read-write. Reset: 0. DC Data error type 1 and poison consumption. MCA_STATUS[Poison] is set on poison consumption from L2/L3.
11	**DcTagErr2**. Read-write. Reset: 0. DC Tag error type 2.
10	**SystemReadDataErrorT1**. Read-write. Reset: 0. Init: BIOS,1. System Read Data Error Thread 1. An error in a read of a line from the data fabric. Possible reasons include master abort and target abort.
9	**SystemReadDataErrorT0**. Read-write. Reset: 0. Init: BIOS,1. System Read Data Error Thread 0. An error in a read of a line from the data fabric. Possible reasons include master abort and target abort.
8	**IntErrTyp2**. Read-write. Reset: 0. Internal error type 2.
7	**IntErrTyp1**. Read-write. Reset: 0. Internal error type 1.
6	**DcTagErr1**. Read-write. Reset: 0. DC Tag error type 1.
5	**DcTagErr6**. Read-write. Reset: 0. DC Tag error type 6.
4	Reserved.
3	**L1DTLB**. Read-write. Reset: 0. Level 1 TLB parity error.
2	**MAB**. Read-write. Reset: 0. Miss address buffer payload parity error.
1	**STQ**. Read-write. Reset: 0. Store queue parity error.
0	**LDQ**. Read-write. Reset: 0. Load queue parity error.

3.2.5.2 IF

MSR0000_0404...MSRC000_2010 [IF Machine Check Control] (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.

_lthree[1:0]_core[3:0]_inst1_aliasMSRLEGACY; MSR0000_0404

_lthree[1:0]_core[3:0]_inst1_aliasMSR; MSRC000_2010

Bits	Description
63:14	Reserved.
13	**SystemReadDataError**. Read-write. Reset: 0. System Read Data Error. An error in a demand fetch of a line. Possible reasons include master abort and target abort.
12	**L2RespPoison**. Read-write. Reset: 0. L2 Cache Response Poison Error. Error is the result of consuming poison data.

11	**L2BtbMultiHit**. Read-write. Reset: 0. L2 BTB Multi-Match Error.
10	**L1BtbMultiHit**. Read-write. Reset: 0. L1 BTB Multi-Match Error.
9	**BpqSnpParT1**. Read-write. Reset: 0. BPQ Thread 1 Snoop Parity Error.
8	**BpqSnpParT0**. Read-write. Reset: 0. BPQ Thread 0 Snoop Parity Error.
7	**L2ItlbParity**. Read-write. Reset: 0. L2 ITLB Parity Error.
6	**L1ItlbParity**. Read-write. Reset: 0. L1 ITLB Parity Error.
5	**L0ItlbParity**. Read-write. Reset: 0. L0 ITLB Parity Error.
4	**DqParity**. Read-write. Reset: 0. Decoupling Queue PhysAddr Parity Error.
3	**DataParity**. Read-write. Reset: 0. IC Data Array Parity Error.
2	**TagParity**. Read-write. Reset: 0. IC Full Tag Parity Error.
1	**TagMultiHit**. Read-write. Reset: 0. IC Microtag or Full Tag Multi-hit Error.
0	**OcUtagParity**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_inst1_aliasMSRLEGACY; MSR0000_0405

_lthree[1:0]_core[3:0]_thread[1:0]_inst1_aliasMSR; MSRC000_2011

Bits	Description
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::IF::MCA_MISC0_IF.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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	**ErrorCode**. 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 37: MCA_STATUS_IF[ErrorCodeExt] Decode

Error Type	Bits	Description
OcUtagParity	[6:0]	0x0
TagMultiHit	[6:0]	0x1
TagParity	[6:0]	0x2
DataParity	[6:0]	0x3
DqParity	[6:0]	0x4
L0ItlbParity	[6:0]	0x5
L1ItlbParity	[6:0]	0x6
L2ItlbParity	[6:0]	0x7
BpqSnpParT0	[6:0]	0x8
BpqSnpParT1	[6:0]	0x9
L1BtbMultiHit	[6:0]	0xa
L2BtbMultiHit	[6:0]	0xb
L2RespPoison	[6:0]	0xc
SystemReadDataError	[6:0]	0xd

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.	
_lthree[1:0]_core[3:0]_thread[1:0]_inst1_aliasMSRLEGACY; MSR0000_0406	
_lthree[1:0]_core[3:0]_thread[1:0]_inst1_aliasMSR; MSRC000_2012	
Bits	**Description**
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. 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[55:0] contains a valid byte address. A value of 6 indicates that MCA::IF::MCA_ADDR_IF[55:6] 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[55:12] contain a valid 4KB memory page and that MCA::IF::MCA_ADDR_IF[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. 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::IF::MCA_STATUS_IF. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 38: MCA_ADDR_IF Register

Error Type	Bits	Description
OcUtagParity	[55:0]	Reserved
TagMultiHit	[55:48] [47:0]	Reserved Physical Address
TagParity	[55:48] [47:0]	Reserved Physical Address
DataParity	[55:48] [47:0]	Reserved Physical Address
DqParity	[55:48] [47:0]	Reserved Physical Address
L0ItlbParity	[55:48] [47:12] [11:0]	Reserved Linear Address Reserved
L1ItlbParity	[55:48] [47:12] [11:0]	Reserved Linear Address Reserved
L2ItlbParity	[55:48] [47:12] [11:0]	Reserved Linear Address Reserved
BpqSnpParT0	[55:0]	Reserved
BpqSnpParT1	[55:0]	Reserved
L1BtbMultiHit	[55:0]	Reserved
L2BtbMultiHit	[55:0]	Reserved
L2RespPoison	[55:48] [47:5] [4:0]	Reserved Physical Address Reserved
SystemReadDataError	[55:48] [47:5] [4:0]	Reserved Physical Address Reserved

MSR0000_0407...MSRC000_2013 [IF Machine Check Miscellaneous 0 Thread 0] (MCA::IF::MCA_MISC0_IF)

Log miscellaneous information associated with errors.
_lthree[1:0]_core[3:0]_thread[1:0]_inst1_aliasMSRLEGACY; MSR0000_0407

_lthree[1:0]_core[3:0]_thread[1:0]_inst1_aliasMSR; MSRC000_2013

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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	**LvtOffset**. 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	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::IF::MCA_MISC0_IF[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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::IF::MCA_MISC0_IF[Locked]) ? Read-write : Read-only.
48	**Ovrflw**. 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	**ErrCnt**. 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	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2014 [IF Machine Check Configuration] (MCA::IF::MCA_CONFIG_IF)

Reset: 0000_0000_0000_0021h.

Controls configuration of the associated machine check bank.

_lthree[1:0]_core[3:0]_inst1_aliasMSR; MSRC000_2014

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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. MCA_DESTAT and MCA_DEADDR are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	**McaX**. 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] (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.

_lthree[1:0]_core[3:0]_inst1_aliasMSR; MSRC000_2015

Bits	Description
63:48	**McaType**. Read-only. Reset: 0001h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. 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

_lthree[1:0]_core[3:0]_thread[1:0]_inst1_aliasMSR; MSRC000_2016

Bits	Description
63:33	Reserved.
32	**Syndrome**. Read-write,Volatile. Reset: Cold,0. 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	**ErrorPriority**. 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	**Length**. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 39 [MCA_SYND_IF Register].

Table 39: MCA_SYND_IF Register

Error Type	Bits	Description
OcUtagParity	[17:5] [4:0]	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
L0ItlbParity	[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	[17:0]	Reserved
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

MSRC001_0401 [IF Machine Check Control Mask] (MCA::IF::MCA_CTL_MASK_IF)

Read-write. Reset: 0000_0000_0000_0000h.

Inhibit detection of an error source.

_lthree[1:0]_core[3:0]_inst1_aliasMSR; MSRC001_0401

Bits	Description
63:14	Reserved.
13	**SystemReadDataError**. Read-write. Reset: 0. Init: BIOS,1. System Read Data Error. An error in a demand fetch of a line. Possible reasons include master abort and target abort.
12	**L2RespPoison**. Read-write. Reset: 0. L2 Cache Response Poison Error. Error is the result of consuming poison data.
11	**L2BtbMultiHit**. Read-write. Reset: 0. L2 BTB Multi-Match Error.
10	**L1BtbMultiHit**. Read-write. Reset: 0. L1 BTB Multi-Match Error.
9	**BpqSnpParT1**. Read-write. Reset: 0. BPQ Thread 1 Snoop Parity Error.
8	**BpqSnpParT0**. Read-write. Reset: 0. BPQ Thread 0 Snoop Parity Error.
7	**L2ItlbParity**. Read-write. Reset: 0. L2 ITLB Parity Error.
6	**L1ItlbParity**. Read-write. Reset: 0. L1 ITLB Parity Error.
5	**L0ItlbParity**. Read-write. Reset: 0. L0 ITLB Parity Error.
4	**DqParity**. Read-write. Reset: 0. Decoupling Queue PhysAddr Parity Error.
3	**DataParity**. Read-write. Reset: 0. IC Data Array Parity Error.
2	**TagParity**. Read-write. Reset: 0. IC Full Tag Parity Error.
1	**TagMultiHit**. Read-write. Reset: 0. IC Microtag or Full Tag Multi-hit Error.

0	**OcUtagParity**. Read-write. Reset: 0. Op Cache Microtag Probe Port Parity Error.

3.2.5.3 L2

MSR0000_0408...MSRC000_2020 [L2 Machine Check Control] (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.	
_lthree[1:0]_core[3:0]_inst2_aliasMSRLEGACY; MSR0000_0408	
_lthree[1:0]_core[3:0]_inst2_aliasMSR; MSRC000_2020	
Bits	**Description**
63:4	Reserved.
3	**Hwa**. Read-write. Reset: 0. Hardware Assert Error.
2	**Data**. Read-write. Reset: 0. L2M Data Array ECC Error.
1	**Tag**. Read-write. Reset: 0. L2M Tag or State Array ECC Error.
0	**MultiHit**. 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.	
_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSRLEGACY; MSR0000_0409	
_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSR; MSRC000_2021	
Bits	**Description**
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::L2::MCA_MISC0_L2.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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	**ErrorCode**. 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 40: MCA_STATUS_L2[ErrorCodeExt] Decode

Error Type	Bits	Description
MultiHit	[6:0]	0x0
Tag	[6:0]	0x1
Data	[6:0]	0x2
Hwa	[6:0]	0x3

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.

_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSRLEGACY; MSR0000_040A

_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSR; MSRC000_2022

Bits	Description
63:62	Reserved.
61:56	**LSB**. 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[55:0] contains a valid byte address. A value of 6 indicates that MCA::L2::MCA_ADDR_L2[55:6] 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[55:12] contain a valid 4KB memory page and that MCA::L2::MCA_ADDR_L2[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. 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::L2::MCA_STATUS_L2. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 41: MCA_ADDR_L2 Register

Error Type	Bits	Description
MultiHit	[55:48] [47:6] [5:0]	Reserved Physical Address Reserved
Tag	[55:48] [47:6] [5:0]	Reserved Physical Address Reserved
Data	[55:48] [47:6] [5:0]	Reserved Physical Address Reserved
Hwa	[31:0]	Reserved

MSR0000_040B...MSRC000_2023 [L2 Machine Check Miscellaneous 0 Thread 0] (MCA::L2::MCA_MISC0_L2)

Log miscellaneous information associated with errors.

_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSRLEGACY; MSR0000_040B

_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSR; MSRC000_2023

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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	**LvtOffset**. 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	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::L2::MCA_MISC0_L2[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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	**Ovrflw**. 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	**ErrCnt**. 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	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2024 [L2 Machine Check Configuration] (MCA::L2::MCA_CONFIG_L2)

Reset: 0000_0000_0000_0025h.

Controls configuration of the associated machine check bank.

_lthree[1:0]_core[3:0]_inst2_aliasMSR; MSRC000_2024

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**LogDeferredInMcaStat**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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	**McaX**. 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] (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.	
_lthree[1:0]_core[3:0]_inst2_aliasMSR; MSRC000_2025	
Bits	**Description**
63:48	**McaType**. Read-only. Reset: 0002h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. 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	
_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSR; MSRC000_2026	
Bits	**Description**
63:49	Reserved.
48:32	**Syndrome**. Read-write,Volatile. Reset: Cold,0_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	**ErrorPriority**. 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	**Length**. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 42 [MCA_SYND_L2 Register].

Table 42: MCA_SYND_L2 Register

Error Type	Bits	Description
MultiHit	[17:8] [7:0]	Index One-hot way vector
Tag	[17:13] [12:3] [2:0]	Reserved Index Way
Data	[17:15] [14:5] [4:3] [2:0]	Reserved Index Quarter-line Way
Hwa	[17:0]	Reserved

MSRC000_2028 [L2 Machine Check Deferred Error Status Thread 0] (MCA::L2::MCA_DESTAT_L2)

Read-write,Volatile. Reset: Cold,0000_0000_0000_0000h.
Holds status information for the first deferred error seen in this bank.
_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSR; MSRC000_2028

Bits	Description
63	**Val**. Read-write,Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	**Overflow**. 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	Reserved.
58	**AddrV**. Read-write,Volatile. Reset: Cold,0. 1=MCA::L2::MCA_DEADDR_L2 contains address information associated with the error.
57:54	Reserved.
53	**SyndV**. 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	Reserved.
44	**Deferred**. 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.
43:0	Reserved.

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_DEADDR_L2[LSB].

_lthree[1:0]_core[3:0]_thread[1:0]_inst2_aliasMSR; MSRC000_2029

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::L2::MCA_DEADDR_L2[ErrorAddr]. A value of 0 indicates that MCA::L2::MCA_DEADDR_L2[55:0] contains a valid byte address. A value of 6 indicates that MCA::L2::MCA_DEADDR_L2[55:6] contains a valid cache line address and that MCA::L2::MCA_DEADDR_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_DEADDR_L2[55:12] contain a valid 4KB memory page and that MCA::L2::MCA_DEADDR_L2[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. Read-write,Volatile. Reset: Cold,00_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_DEADDR_L2[LSB].

MSRC001_0402 [**L2 Machine Check Control Mask**] **(MCA::L2::MCA_CTL_MASK_L2)**

Read-write. Reset: 0000_0000_0000_0000h.

Inhibit detection of an error source.

_lthree[1:0]_core[3:0]_inst2_aliasMSR; MSRC001_0402

Bits	Description
63:4	Reserved.
3	**Hwa**. Read-write. Reset: 0. Init: BIOS,1. Hardware Assert Error.
2	**Data**. Read-write. Reset: 0. L2M Data Array ECC Error.
1	**Tag**. Read-write. Reset: 0. L2M Tag or State Array ECC Error.
0	**MultiHit**. Read-write. Reset: 0. L2M Tag Multiple-Way-Hit error.

3.2.5.4 DE

MSR0000_040C...MSRC000_2030 [DE Machine Check Control] (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.

_lthree[1:0]_core[3:0]_inst3_aliasMSRLEGACY; MSR0000_040C

_lthree[1:0]_core[3:0]_inst3_aliasMSR; MSRC000_2030

Bits	Description
63:9	Reserved.
8	**OCBQ**. Read-write. Reset: 0. Micro-op buffer parity error.
7	**UcSeq**. Read-write. Reset: 0. Patch RAM sequencer parity error.
6	**UcDat**. Read-write. Reset: 0. Patch RAM data parity error.
5	**Faq**. Read-write. Reset: 0. Fetch address FIFO parity error.
4	**Idq**. Read-write. Reset: 0. Instruction dispatch queue parity error.
3	**UopQ**. Read-write. Reset: 0. Micro-op queue parity error.
2	**Ibq**. Read-write. Reset: 0. Instruction buffer parity error.
1	**OcDat**. Read-write. Reset: 0. Micro-op cache data parity error.
0	**OcTag**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_inst3_aliasMSRLEGACY; MSR0000_040D

_lthree[1:0]_core[3:0]_thread[1:0]_inst3_aliasMSR; MSRC000_2031

Bits	Description
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::DE::MCA_MISC0_DE.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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	**ErrorCode**. 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 43: MCA_STATUS_DE[ErrorCodeExt] Decode

Error Type	Bits	Description
OcTag	[6:0]	0x0
OcDat	[6:0]	0x1
Ibq	[6:0]	0x2
UopQ	[6:0]	0x3
Idq	[6:0]	0x4
Faq	[6:0]	0x5
UcDat	[6:0]	0x6
UcSeq	[6:0]	0x7
OCBQ	[6:0]	0x8

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.

_lthree[1:0]_core[3:0]_thread[1:0]_inst3_aliasMSRLEGACY; MSR0000_040E

_lthree[1:0]_core[3:0]_thread[1:0]_inst3_aliasMSR; MSRC000_2032

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. 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[55:0] contains a valid byte address. A value of 6 indicates that MCA::DE::MCA_ADDR_DE[55:6] 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[55:12] contain a valid 4KB memory page and that MCA::DE::MCA_ADDR_DE[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. 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::DE::MCA_STATUS_DE. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 44: MCA_ADDR_DE Register

Error Type	Bits	Description
OcTag	[55:0]	Reserved
OcDat	[55:0]	Reserved
Ibq	[55:0]	Reserved
UopQ	[55:0]	Reserved
Idq	[55:0]	Reserved
Faq	[55:0]	Reserved
UcDat	[55:0]	Reserved
UcSeq	[55:0]	Reserved
OCBQ	[55:0]	Reserved

MSR0000_040F...MSRC000_2033 [DE Machine Check Miscellaneous 0 Thread 0] (MCA::DE::MCA_MISC0_DE)

Log miscellaneous information associated with errors.

_lthree[1:0]_core[3:0]_thread[1:0]_inst3_aliasMSRLEGACY; MSR0000_040F

_lthree[1:0]_core[3:0]_thread[1:0]_inst3_aliasMSR; MSRC000_2033

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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	**LvtOffset**. 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	**CntEn**. Reset: 0. 1=Count thresholding errors.

	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::DE::MCA_MISC0_DE[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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	**Ovrflw**. 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	**ErrCnt**. 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	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2034 [DE Machine Check Configuration] (MCA::DE::MCA_CONFIG_DE)

Reset: 0000_0000_0000_0021h.

Controls configuration of the associated machine check bank.

_lthree[1:0]_core[3:0]_inst3_aliasMSR; MSRC000_2034

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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. MCA_DESTAT and MCA_DEADDR are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	**McaX**. 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] (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.	
_lthree[1:0]_core[3:0]_inst3_aliasMSR; MSRC000_2035	
Bits	**Description**
63:48	**McaType**. Read-only. Reset: 0003h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. 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	
_lthree[1:0]_core[3:0]_thread[1:0]_inst3_aliasMSR; MSRC000_2036	
Bits	**Description**
63:33	Reserved.
32	**Syndrome**. Read-write,Volatile. Reset: Cold,0. 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	**ErrorPriority**. 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	**Length**. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 45 [MCA_SYND_DE Register].

Table 45: MCA_SYND_DE Register

Error Type	Bits	Description
OcTag	[17:16] [15:8] [7:0]	Reserved Index Way
OcDat	[17:16] [15:8] [7:0]	Reserved Index Way
Ibq	[17:0]	Reserved
UopQ	[17:0]	Reserved
Idq	[17:0]	Reserved
Faq	[17:0]	Reserved
UcDat	[17:0]	Reserved
UcSeq	[17:0]	Reserved
OCBQ	[17:0]	Reserved

MSRC001_0403 [DE Machine Check Control Mask] (MCA::DE::MCA_CTL_MASK_DE)

Read-write. Reset: 0000_0000_0000_0000h.

Inhibit detection of an error source.

_lthree[1:0]_core[3:0]_inst3_aliasMSR; MSRC001_0403

Bits	Description
63:9	Reserved.
8	**OCBQ**. Read-write. Reset: 0. Micro-op buffer parity error.
7	**UcSeq**. Read-write. Reset: 0. Patch RAM sequencer parity error.
6	**UcDat**. Read-write. Reset: 0. Patch RAM data parity error.
5	**Faq**. Read-write. Reset: 0. Fetch address FIFO parity error.
4	**Idq**. Read-write. Reset: 0. Instruction dispatch queue parity error.
3	**UopQ**. Read-write. Reset: 0. Micro-op queue parity error.
2	**Ibq**. Read-write. Reset: 0. Instruction buffer parity error.
1	**OcDat**. Read-write. Reset: 0. Micro-op cache data parity error.
0	**OcTag**. Read-write. Reset: 0. Micro-op cache tag parity error.

3.2.5.5 EX

MSR0000_0414...MSRC000_2050 [**EX Machine Check Control**] **(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.

_lthree[1:0]_core[3:0]_inst5_aliasMSRLEGACY; MSR0000_0414

_lthree[1:0]_core[3:0]_inst5_aliasMSR; MSRC000_2050

Bits	Description
63:11	Reserved.
10	**BBQ**. Read-write. Reset: 0. Branch buffer queue parity error.
9	**SQ**. Read-write. Reset: 0. Scheduling queue parity error.
8	**STATQ**. Read-write. Reset: 0. Retire status queue parity error.
7	**RETDISP**. Read-write. Reset: 0. Retire dispatch queue parity error.
6	**CHKPTQ**. Read-write. Reset: 0. CHKPTQ. Checkpoint queue parity error.
5	**PLDAL**. Read-write. Reset: 0. EX payload parity error.
4	**PLDAG**. Read-write. Reset: 0. Address generator payload parity error.
3	**IDRF**. Read-write. Reset: 0. Immediate displacement register file parity error.
2	**FRF**. Read-write. Reset: 0. Flag register file parity error.
1	**PRF**. Read-write. Reset: 0. Physical register file parity error.
0	**WDT**. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_inst5_aliasMSRLEGACY; MSR0000_0415

_lthree[1:0]_core[3:0]_thread[1:0]_inst5_aliasMSR; MSRC000_2051

Bits	Description
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::EX::MCA_MISC0_EX.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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	**ErrorCode**. 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 46: MCA_STATUS_EX[ErrorCodeExt] Decode

Error Type	Bits	Description
WDT	[6:0]	0x0
PRF	[6:0]	0x1
FRF	[6:0]	0x2
IDRF	[6:0]	0x3
PLDAG	[6:0]	0x4
PLDAL	[6:0]	0x5
CHKPTQ	[6:0]	0x6
RETDISP	[6:0]	0x7
STATQ	[6:0]	0x8
SQ	[6:0]	0x9
BBQ	[6:0]	0xa

MSR0000_0416...MSRC000_2052 [**EX Machine Check Address Thread 0**] **(MCA::EX::MCA_ADDR_EX)**

Read-only. 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.

_lthree[1:0]_core[3:0]_thread[1:0]_inst5_aliasMSRLEGACY; MSR0000_0416

_lthree[1:0]_core[3:0]_thread[1:0]_inst5_aliasMSR; MSRC000_2052

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-only. 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[55:0] contains a valid byte address. A value of 6 indicates that MCA::EX::MCA_ADDR_EX[55:6] 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[55:12] contain a valid 4KB memory page and that MCA::EX::MCA_ADDR_EX[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. Read-only. Reset: Cold,00_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::EX::MCA_STATUS_EX.

Table 47: MCA_ADDR_EX Register

Error Type	Bits	Description
WDT	55:49] [48:0]	Reserved RIP of thread triggering the watchdog timeout
PRF	[55:0]	Reserved
FRF	[55:0]	Reserved
IDRF	[55:0]	Reserved
PLDAG	[55:0]	Reserved
PLDAL	[55:0]	Reserved
CHKPTQ	[55:0]	Reserved
RETDISP	[55:0]	Reserved
STATQ	[55:0]	Reserved
SQ	[55:0]	Reserved
BBQ	[55:0]	Reserved

MSR0000_0417...MSRC000_2053 [EX Machine Check Miscellaneous 0 Thread 0] (MCA::EX::MCA_MISC0_EX)

Log miscellaneous information associated with errors.

_lthree[1:0]_core[3:0]_thread[1:0]_inst5_aliasMSRLEGACY; MSR0000_0417

_lthree[1:0]_core[3:0]_thread[1:0]_inst5_aliasMSR; MSRC000_2053

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	**LvtOffset**. 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::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
51	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
48	**Ovrflw**. 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::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	**ErrCnt**. 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::EX::MCA_MISC0_EX[Locked]) ? Read-write : Read-only.
31:24	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2054 [EX Machine Check Configuration] (MCA::EX::MCA_CONFIG_EX)

Reset: 0000_0000_0000_0021h.

Controls configuration of the associated machine check bank.

_lthree[1:0]_core[3:0]_inst5_aliasMSR; MSRC000_2054

Bits	Description

63:39	Reserved.
38:37	**DeferredIntType**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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. MCA_DESTAT and MCA_DEADDR are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	**McaX**. 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] (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.

_lthree[1:0]_core[3:0]_inst5_aliasMSR; MSRC000_2055

Bits	Description
63:48	**McaType**. Read-only. Reset: 0005h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. 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

_lthree[1:0]_core[3:0]_thread[1:0]_inst5_aliasMSR; MSRC000_2056

Bits	Description
63:33	Reserved.
32	**Syndrome**. Read-write,Volatile. Reset: Cold,0. 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	**ErrorPriority**. 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	**Length**. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 48 [MCA_SYND_EX Register].

Table 48: MCA_SYND_EX Register

Error Type	Bits	Description
WDT	[17:0]	Reserved
PRF	[17:0]	Reserved
FRF	[17:4] [3:0]	Reserved Reserved
IDRF	[17:6] [5:4] [3:0]	Reserved Reserved Reserved
PLDAG	[17:2] [1:0]	Reserved Reserved
PLDAL	[17:4] [3:0]	Reserved Reserved
CHKPTQ	[17:4] [3:2] [1:0]	Reserved Reserved Reserved
RETDISP	[17:2] [1:0]	Reserved Reserved
STATQ	[17:0]	Reserved
SQ	[17:6] [5:0]	Reserved Reserved
BBQ	[17:6] [5:0]	Reserved Reserved

MSRC001_0405 [EX Machine Check Control Mask] (MCA::EX::MCA_CTL_MASK_EX)

Read-write. Reset: 0000_0000_0000_0000h.

Inhibit detection of an error source.

_lthree[1:0]_core[3:0]_inst5_aliasMSR; MSRC001_0405

Bits	Description
63:11	Reserved.
10	**BBQ**. Read-write. Reset: 0. Branch buffer queue parity error.
9	**SQ**. Read-write. Reset: 0. Scheduling queue parity error.
8	**STATQ**. Read-write. Reset: 0. Retire status queue parity error.
7	**RETDISP**. Read-write. Reset: 0. Retire dispatch queue parity error.
6	**CHKPTQ**. Read-write. Reset: 0. CHKPTQ. Checkpoint queue parity error.
5	**PLDAL**. Read-write. Reset: 0. EX payload parity error.
4	**PLDAG**. Read-write. Reset: 0. Address generator payload parity error.
3	**IDRF**. Read-write. Reset: 0. Immediate displacement register file parity error.
2	**FRF**. Read-write. Reset: 0. Flag register file parity error.
1	**PRF**. Read-write. Reset: 0. Physical register file parity error.
0	**WDT**. Read-write. Reset: 0. Watchdog Timeout error.

3.2.5.6 FP

MSR0000_0418...MSRC000_2060 [FP Machine Check Control] (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.	
_lthree[1:0]_core[3:0]_inst6_aliasMSRLEGACY; MSR0000_0418	
_lthree[1:0]_core[3:0]_inst6_aliasMSR; MSRC000_2060	
Bits	**Description**
63:7	Reserved.
6	**HWA**. Read-write. Reset: 0. Hardware assertion.
5	**SRF**. Read-write. Reset: 0. Status register file (SRF) parity error.
4	**RQ**. Read-write. Reset: 0. Retire queue (RQ) parity error.
3	**NSQ**. Read-write. Reset: 0. NSQ parity error.
2	**SCH**. Read-write. Reset: 0. Schedule queue parity error.
1	**FL**. Read-write. Reset: 0. Freelist (FL) parity error.
0	**PRF**. 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.	
_lthree[1:0]_core[3:0]_thread[1:0]_inst6_aliasMSRLEGACY; MSR0000_0419	
_lthree[1:0]_core[3:0]_thread[1:0]_inst6_aliasMSR; MSRC000_2061	
Bits	**Description**
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::FP::MCA_MISC0_FP.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
15:0	**ErrorCode**. 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 49: MCA_STATUS_FP[ErrorCodeExt] Decode

Error Type	Bits	Description
PRF	[6:0]	0x0
FL	[6:0]	0x1
SCH	[6:0]	0x2
NSQ	[6:0]	0x3
RQ	[6:0]	0x4
SRF	[6:0]	0x5
HWA	[6:0]	0x6

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.
_lthree[1:0]_core[3:0]_thread[1:0]_inst6_aliasMSRLEGACY; MSR0000_041A
_lthree[1:0]_core[3:0]_thread[1:0]_inst6_aliasMSR; MSRC000_2062

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-only. 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[55:0] contains a valid byte address. A value of 6 indicates that MCA::FP::MCA_ADDR_FP[55:6] 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[55:12] contain a valid 4KB memory page and that MCA::FP::MCA_ADDR_FP[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. Read-only. Reset: Cold,00_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::FP::MCA_STATUS_FP.

Table 50: MCA_ADDR_FP Register

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

MSR0000_041B...MSRC000_2063 [**FP Machine Check Miscellaneous 0 Thread 0**] **(MCA::FP::MCA_MISC0_FP)**

Log miscellaneous information associated with errors.

_lthree[1:0]_core[3:0]_thread[1:0]_inst6_aliasMSRLEGACY; MSR0000_041B

_lthree[1:0]_core[3:0]_thread[1:0]_inst6_aliasMSR; MSRC000_2063

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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	**LvtOffset**. 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	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::FP::MCA_MISC0_FP[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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::FP::MCA_MISC0_FP[Locked]) ? Read-write :

	Read-only.
48	**Ovrflw**. 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	**ErrCnt**. 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	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2064 [FP Machine Check Configuration] (MCA::FP::MCA_CONFIG_FP)

Reset: 0000_0000_0000_0021h.

Controls configuration of the associated machine check bank.

_lthree[1:0]_core[3:0]_inst6_aliasMSR; MSRC000_2064

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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. MCA_DESTAT and MCA_DEADDR are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	**McaX**. 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] (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.

_lthree[1:0]_core[3:0]_inst6_aliasMSR; MSRC000_2065

Bits	Description
63:48	**McaType**. Read-only. Reset: 0006h. The McaType of the MCA bank within this IP.

47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. 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

_lthree[1:0]_core[3:0]_thread[1:0]_inst6_aliasMSR; MSRC000_2066

Bits	Description
63:33	Reserved.
32	**Syndrome**. Read-write,Volatile. Reset: Cold,0. Contains the syndrome, if any, associated with the error logged in MCA::FP::MCA_STATUS_FP. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::FP::MCA_SYND_FP[Length]. The Syndrome field is only valid when MCA::FP::MCA_SYND_FP[Length] is not 0.
31:27	Reserved.
26:24	**ErrorPriority**. 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	**Length**. Read-write,Volatile. Reset: Cold,00h. Specifies the length in bits of the syndrome contained in MCA::FP::MCA_SYND_FP[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::FP::MCA_SYND_FP. For example, a syndrome length of 9 means that MCA::FP::MCA_SYND_FP[Syndrome] bits [8:0] contains a valid syndrome.
17:0	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 51 [MCA_SYND_FP Register].

Table 51: MCA_SYND_FP Register

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

MSRC001_0406 [FP Machine Check Control Mask] (MCA::FP::MCA_CTL_MASK_FP)

Read-write. Reset: 0000_0000_0000_0000h.

Inhibit detection of an error source.

_lthree[1:0]_core[3:0]_inst6_aliasMSR; MSRC001_0406

Bits	Description
63:7	Reserved.
6	**HWA**. Read-write. Reset: 0. Init: BIOS,1. Hardware assertion.
5	**SRF**. Read-write. Reset: 0. Status register file (SRF) parity error.
4	**RQ**. Read-write. Reset: 0. Retire queue (RQ) parity error.
3	**NSQ**. Read-write. Reset: 0. NSQ parity error.
2	**SCH**. Read-write. Reset: 0. Schedule queue parity error.
1	**FL**. Read-write. Reset: 0. Freelist (FL) parity error.
0	**PRF**. Read-write. Reset: 0. Physical register file (PRF) parity error.

3.2.5.7 L3 Cache

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.	
_ccx0_inst7_aliasMSRLEGACY; MSR0000_041C	
_ccx0_inst8_aliasMSRLEGACY; MSR0000_0420	
_ccx0_inst9_aliasMSRLEGACY; MSR0000_0424	
_ccx0_inst10_aliasMSRLEGACY; MSR0000_0428	
_ccx1_inst7_aliasMSRLEGACY; MSR0000_042C	
_ccx1_inst8_aliasMSRLEGACY; MSR0000_0430	
_ccx1_inst9_aliasMSRLEGACY; MSR0000_0434	
_ccx1_inst10_aliasMSRLEGACY; MSR0000_0438	
_ccx0_inst7_aliasMSR; MSRC000_2070	
_ccx0_inst8_aliasMSR; MSRC000_2080	
_ccx0_inst9_aliasMSR; MSRC000_2090	
_ccx0_inst10_aliasMSR; MSRC000_20A0	
_ccx1_inst7_aliasMSR; MSRC000_20B0	
_ccx1_inst8_aliasMSR; MSRC000_20C0	
_ccx1_inst9_aliasMSR; MSRC000_20D0	
_ccx1_inst10_aliasMSR; MSRC000_20E0	
Bits	**Description**
63:8	Reserved.
7	**Hwa**. Read-write. Reset: 0. L3 Hardware Assertion.
6	**XiVictimQueue**. Read-write. Reset: 0. L3 Victim Queue Parity Error.
5	**SdpParity**. Read-write. Reset: 0. SDP Parity Error from XI.
4	**DataArray**. Read-write. Reset: 0. L3M Data ECC Error.
3	**MultiHitTag**. Read-write. Reset: 0. L3M Tag Multi-way-hit Error.
2	**Tag**. Read-write. Reset: 0. L3M Tag ECC Error.
1	**MultiHitShadowTag**. Read-write. Reset: 0. Shadow Tag Macro Multi-way-hit Error.
0	**ShadowTag**. 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.	
_ccx0_inst7_aliasMSRLEGACY; MSR0000_041D	
_ccx0_inst8_aliasMSRLEGACY; MSR0000_0421	
_ccx0_inst9_aliasMSRLEGACY; MSR0000_0425	
_ccx0_inst10_aliasMSRLEGACY; MSR0000_0429	
_ccx1_inst7_aliasMSRLEGACY; MSR0000_042D	
_ccx1_inst8_aliasMSRLEGACY; MSR0000_0431	
_ccx1_inst9_aliasMSRLEGACY; MSR0000_0435	
_ccx1_inst10_aliasMSRLEGACY; MSR0000_0439	
_ccx0_inst7_aliasMSR; MSRC000_2071	
_ccx0_inst8_aliasMSR; MSRC000_2081	
_ccx0_inst9_aliasMSR; MSRC000_2091	
_ccx0_inst10_aliasMSR; MSRC000_20A1	
_ccx1_inst7_aliasMSR; MSRC000_20B1	
_ccx1_inst8_aliasMSR; MSRC000_20C1	
_ccx1_inst9_aliasMSR; MSRC000_20D1	
_ccx1_inst10_aliasMSR; MSRC000_20E1	
Bits	**Description**
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::L3::MCA_MISC0_L3.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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	**ErrorCode**. 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 52: MCA_STATUS_L3[ErrorCodeExt] Decode

Error Type	Bits	Description
ShadowTag	[6:0]	0x0
MultiHitShadowTag	[6:0]	0x1
Tag	[6:0]	0x2
MultiHitTag	[6:0]	0x3
DataArray	[6:0]	0x4
SdpParity	[6:0]	0x5
XiVictimQueue	[6:0]	0x6
Hwa	[6:0]	0x7

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.

_ccx0_inst7_aliasMSRLEGACY; MSR0000_041E
_ccx0_inst8_aliasMSRLEGACY; MSR0000_0422
_ccx0_inst9_aliasMSRLEGACY; MSR0000_0426
_ccx0_inst10_aliasMSRLEGACY; MSR0000_042A
_ccx1_inst7_aliasMSRLEGACY; MSR0000_042E
_ccx1_inst8_aliasMSRLEGACY; MSR0000_0432
_ccx1_inst9_aliasMSRLEGACY; MSR0000_0436
_ccx1_inst10_aliasMSRLEGACY; MSR0000_043A
_ccx0_inst7_aliasMSR; MSRC000_2072
_ccx0_inst8_aliasMSR; MSRC000_2082
_ccx0_inst9_aliasMSR; MSRC000_2092
_ccx0_inst10_aliasMSR; MSRC000_20A2
_ccx1_inst7_aliasMSR; MSRC000_20B2
_ccx1_inst8_aliasMSR; MSRC000_20C2
_ccx1_inst9_aliasMSR; MSRC000_20D2
_ccx1_inst10_aliasMSR; MSRC000_20E2

Bits	Description
63:62	Reserved.
61:56	**LSB**. 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[55:0] contains a valid byte address. A value of 6 indicates that MCA::L3::MCA_ADDR_L3[55:6] 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[55:12] contain a valid 4KB memory page and that MCA::L3::MCA_ADDR_L3[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. 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::L3::MCA_STATUS_L3. For physical addresses, the most significant bit is given by Core::X86::Cpuid::LongModeInfo[PhysAddrSize].

Table 53: MCA_ADDR_L3 Register

Error Type	Bits	Description
ShadowTag	[55:16]	Reserved

	[15:0]	16'b{8'b{Index}, 2'b{Slice}, 6'b{0}}
MultiHitShadowTag	[55:16] [15:0]	Reserved 16'b{8'b{Index}, 2'b{Slice}, 6'b{0}}
Tag	[55:19] [18:0]	Reserved 19'b{1'b{Bank[3]}, 7'b{Index}, 3'b{Bank[2:0]}, 2'b{slice}, 6'b{0}}
MultiHitTag	[55:19] [18:0]	Reserved 19'b{1'b{Bank[3]}, 7'b{Index}, 3'b{Bank[2:0]}, 2'b{slice}, 6'b{0}}
DataArray	[55:48] [47:0]	Reserved Physical Address
SdpParity	[55:48] [47:0]	Reserved Physical Address
XiVictimQueue	[55:48] [47:0]	Reserved Physical Address
Hwa	[55:34] [33:0]	Reserved Reserved

MSR0000_041F...MSRC000_20E3 [L3 Machine Check Miscellaneous 0] (MCA::L3::MCA_MISC0_L3)

Log miscellaneous information associated with errors.

_ccx0_inst7_aliasMSRLEGACY; MSR0000_041F
_ccx0_inst8_aliasMSRLEGACY; MSR0000_0423
_ccx0_inst9_aliasMSRLEGACY; MSR0000_0427
_ccx0_inst10_aliasMSRLEGACY; MSR0000_042B
_ccx1_inst7_aliasMSRLEGACY; MSR0000_042F
_ccx1_inst8_aliasMSRLEGACY; MSR0000_0433
_ccx1_inst9_aliasMSRLEGACY; MSR0000_0437
_ccx1_inst10_aliasMSRLEGACY; MSR0000_043B
_ccx0_inst7_aliasMSR; MSRC000_2073
_ccx0_inst8_aliasMSR; MSRC000_2083
_ccx0_inst9_aliasMSR; MSRC000_2093
_ccx0_inst10_aliasMSR; MSRC000_20A3
_ccx1_inst7_aliasMSR; MSRC000_20B3
_ccx1_inst8_aliasMSR; MSRC000_20C3
_ccx1_inst9_aliasMSR; MSRC000_20D3
_ccx1_inst10_aliasMSR; MSRC000_20E3

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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	**LvtOffset**. 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	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::L3::MCA_MISC0_L3[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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	**Ovrflw**. 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	**ErrCnt**. 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	**BlkPtr**. Read-write. Reset: 01h. 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_0025h.

Controls configuration of the associated machine check bank.

_ccx0_inst7_aliasMSR; MSRC000_2074

_ccx0_inst8_aliasMSR; MSRC000_2084

_ccx0_inst9_aliasMSR; MSRC000_2094

_ccx0_inst10_aliasMSR; MSRC000_20A4

_ccx1_inst7_aliasMSR; MSRC000_20B4

_ccx1_inst8_aliasMSR; MSRC000_20C4

_ccx1_inst9_aliasMSR; MSRC000_20D4

_ccx1_inst10_aliasMSR; MSRC000_20E4

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**LogDeferredInMcaStat**. 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	**McaXEnable**. 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.
31:6	Reserved.

5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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	**McaX**. 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.

_ccx0_inst7_aliasMSR; MSRC000_2075
_ccx0_inst8_aliasMSR; MSRC000_2085
_ccx0_inst9_aliasMSR; MSRC000_2095
_ccx0_inst10_aliasMSR; MSRC000_20A5
_ccx1_inst7_aliasMSR; MSRC000_20B5
_ccx1_inst8_aliasMSR; MSRC000_20C5
_ccx1_inst9_aliasMSR; MSRC000_20D5
_ccx1_inst10_aliasMSR; MSRC000_20E5

Bits	Description
63:48	**McaType**. Read-only. Reset: 0007h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 0B0h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. 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

_ccx0_inst7_aliasMSR; MSRC000_2076
_ccx0_inst8_aliasMSR; MSRC000_2086
_ccx0_inst9_aliasMSR; MSRC000_2096
_ccx0_inst10_aliasMSR; MSRC000_20A6
_ccx1_inst7_aliasMSR; MSRC000_20B6
_ccx1_inst8_aliasMSR; MSRC000_20C6
_ccx1_inst9_aliasMSR; MSRC000_20D6
_ccx1_inst10_aliasMSR; MSRC000_20E6

Bits	Description
63:49	Reserved.
48:32	**Syndrome**. Read-write,Volatile. Reset: Cold,0_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	**ErrorPriority**. 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	**Length**. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 54 [MCA_SYND_L3 Register].

Table 54: MCA_SYND_L3 Register

Error Type	Bits	Description
ShadowTag	[17:12] [11:8] [7:3] [2:0]	Reserved Pack Reserved Way
MultiHitShadowTag	[17:12] [11:8] [7:0]	Reserved Pack Reserved
Tag	[17:12] [11:8] [7:0]	Reserved Bank. Way
MultiHitTag	[17:0]	Reserved
DataArray	[17:12] [11:8] [7:3] [2:0]	Reserved Bank[2:0] Reserved Way
SdpParity	[17:0]	Reserved
XiVictimQueue	[17:0]	Reserved
Hwa	[17:0]	Reserved

MSRC000_20[7...E]8 [L3 Machine Check Deferred Error Status] (MCA::L3::MCA_DESTAT_L3)

Read-write,Volatile. Reset: Cold,0000_0000_0000_0000h.

Holds status information for the first deferred error seen in this bank.

_ccx0_inst7_aliasMSR; MSRC000_2078
_ccx0_inst8_aliasMSR; MSRC000_2088
_ccx0_inst9_aliasMSR; MSRC000_2098
_ccx0_inst10_aliasMSR; MSRC000_20A8
_ccx1_inst7_aliasMSR; MSRC000_20B8
_ccx1_inst8_aliasMSR; MSRC000_20C8
_ccx1_inst9_aliasMSR; MSRC000_20D8
_ccx1_inst10_aliasMSR; MSRC000_20E8

Bits	Description
63	**Val**. Read-write,Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	**Overflow**. 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	Reserved.
58	**AddrV**. Read-write,Volatile. Reset: Cold,0. 1=MCA::L3::MCA_DEADDR_L3 contains address information associated with the error.
57:54	Reserved.
53	**SyndV**. 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	Reserved.
44	**Deferred**. 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.
43:0	Reserved.

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_DEADDR_L3[LSB].

_ccx0_inst7_aliasMSR; MSRC000_2079
_ccx0_inst8_aliasMSR; MSRC000_2089
_ccx0_inst9_aliasMSR; MSRC000_2099
_ccx0_inst10_aliasMSR; MSRC000_20A9
_ccx1_inst7_aliasMSR; MSRC000_20B9
_ccx1_inst8_aliasMSR; MSRC000_20C9
_ccx1_inst9_aliasMSR; MSRC000_20D9
_ccx1_inst10_aliasMSR; MSRC000_20E9

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::L3::MCA_DEADDR_L3[ErrorAddr]. A value of 0 indicates that MCA::L3::MCA_DEADDR_L3[55:0] contains a valid byte address. A value of 6 indicates that MCA::L3::MCA_DEADDR_L3[55:6] contains a valid cache line address and that MCA::L3::MCA_DEADDR_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_DEADDR_L3[55:12] contain a valid 4KB memory page and that MCA::L3::MCA_DEADDR_L3[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. Read-write,Volatile. Reset: Cold,00_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_DEADDR_L3[LSB].

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.

_ccx0_inst7_aliasMSR; MSRC001_0407
_ccx0_inst8_aliasMSR; MSRC001_0408
_ccx0_inst9_aliasMSR; MSRC001_0409
_ccx0_inst10_aliasMSR; MSRC001_040A
_ccx1_inst7_aliasMSR; MSRC001_040B
_ccx1_inst8_aliasMSR; MSRC001_040C
_ccx1_inst9_aliasMSR; MSRC001_040D
_ccx1_inst10_aliasMSR; MSRC001_040E

Bits	Description
63:8	Reserved.
7	**Hwa**. Read-write. Reset: 0. Init: BIOS,1. L3 Hardware Assertion.
6	**XiVictimQueue**. Read-write. Reset: 0. L3 Victim Queue Parity Error.
5	**SdpParity**. Read-write. Reset: 0. SDP Parity Error from XI.
4	**DataArray**. Read-write. Reset: 0. L3M Data ECC Error.
3	**MultiHitTag**. Read-write. Reset: 0. L3M Tag Multi-way-hit Error.
2	**Tag**. Read-write. Reset: 0. L3M Tag ECC Error.
1	**MultiHitShadowTag**. Read-write. Reset: 0. Shadow Tag Macro Multi-way-hit Error.
0	**ShadowTag**. Read-write. Reset: 0. Shadow Tag Macro ECC Error.

3.2.5.8 CS

MSR0000_0450...MSRC000_2150 [**CS Machine Check Control**] **(MCA::CS::MCA_CTL_CS)**

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.

_instCS0_aliasMSRLEGACY; MSR0000_0450
_instCS1_aliasMSRLEGACY; MSR0000_0454
_instCS0_aliasMSR; MSRC000_2140
_instCS1_aliasMSR; MSRC000_2150

Bits	Description
63:9	Reserved.
8	**SPF_ECC_ERR**. Read-write. Reset: 0. Probe Filter ECC Error: An ECC error occurred on a probe filter access.
7	**ATM_PAR_ERR**. Read-write. Reset: 0. Atomic Request Parity Error: Parity error on read of an atomic transaction.
6	**SDP_PAR_ERR**. Read-write. Reset: 0. Read Response Parity Error: Parity error on incoming read response data.
5	**FTI_PAR_ERR**. Read-write. Reset: 0. Request or Probe Parity Error: Parity error on incoming request or probe response data.
4	**FTI_RSP_NO_MTCH**. Read-write. Reset: 0. Unexpected Response: A response was received from the transport layer which does not match any request.
3	**FTI_ILL_RSP**. Read-write. Reset: 0. Illegal Response: An illegal response was received from the transport layer.
2	**FTI_SEC_VIOL**. Read-write. Reset: 0. Security Violation: A security violation was received from the transport layer.
1	**FTI_ADDR_VIOL**. Read-write. Reset: 0. Address Violation: An address violation was received from the transport layer.
0	**FTI_ILL_REQ**. Read-write. Reset: 0. Illegal Request: An illegal request was received from the transport layer.

MSR0000_0451...MSRC000_2151 [**CS Machine Check Status**] **(MCA::CS::MCA_STATUS_CS)**

Reset: Cold,0000_0000_0000_0000h.

Logs information associated with errors.

_instCS0_aliasMSRLEGACY; MSR0000_0451
_instCS1_aliasMSRLEGACY; MSR0000_0455
_instCS0_aliasMSR; MSRC000_2141
_instCS1_aliasMSR; MSRC000_2151

Bits	Description
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::CS::MCA_MISC0_CS.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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	**ErrorCode**. 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 55: MCA_STATUS_CS[ErrorCodeExt] Decode

Error Type	Bits	Description
FTI_ILL_REQ	[6:0]	0x0
FTI_ADDR_VIOL	[6:0]	0x1
FTI_SEC_VIOL	[6:0]	0x2
FTI_ILL_RSP	[6:0]	0x3

FTI_RSP_NO_MTCH	[6:0]	0x4
FTI_PAR_ERR	[6:0]	0x5
SDP_PAR_ERR	[6:0]	0x6
ATM_PAR_ERR	[6:0]	0x7
SPF_ECC_ERR	[6:0]	0x8

MSR0000_0452...MSRC000_2152 [CS Machine Check Address] (MCA::CS::MCA_ADDR_CS)

Read-write,Volatile. 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_aliasMSRLEGACY; MSR0000_0452
_instCS1_aliasMSRLEGACY; MSR0000_0456
_instCS0_aliasMSR; MSRC000_2142
_instCS1_aliasMSR; MSRC000_2152

Bits	Description
63:62	Reserved.
61:56	**LSB**. 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.
55:0	**ErrorAddr**. 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 56: MCA_ADDR_CS Register

Error Type	Bits	Description
FTI_ILL_REQ	[55:0]	Reserved
FTI_ADDR_VIOL	[55:0]	Reserved
FTI_SEC_VIOL	[55:0]	Reserved
FTI_ILL_RSP	[55:0]	Reserved
FTI_RSP_NO_MTCH	[55:0]	Reserved
FTI_PAR_ERR	[55:0]	Reserved
SDP_PAR_ERR	[55:0]	Reserved
ATM_PAR_ERR	[55:0]	Reserved
SPF_ECC_ERR	[55:0]	Reserved

MSR0000_0453...MSRC000_2153 [CS Machine Check Miscellaneous 0] (MCA::CS::MCA_MISC0_CS)

Log miscellaneous information associated with errors.

_instCS0_aliasMSRLEGACY; MSR0000_0453
_instCS1_aliasMSRLEGACY; MSR0000_0457
_instCS0_aliasMSR; MSRC000_2143
_instCS1_aliasMSR; MSRC000_2153

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	**LvtOffset**. 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::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
51	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
48	**Ovrflw**. 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::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	**ErrCnt**. 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::CS::MCA_MISC0_CS[Locked]) ? Read-write : Read-only.
31:24	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_21[4...5]4 [CS Machine Check Configuration] (MCA::CS::MCA_CONFIG_CS)

Reset: 0000_0000_0000_0025h.

Controls configuration of the associated machine check bank.

_instCS0_aliasMSR; MSRC000_2144

_instCS1_aliasMSR; MSRC000_2154

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**LogDeferredInMcaStat**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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	**McaX**. 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[4...5]5 [CS IP Identification] (MCA::CS::MCA_IPID_CS)

Reset: 0000_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.

_instCS0_aliasMSR; MSRC000_2145

_instCS1_aliasMSR; MSRC000_2155

Bits	Description
63:48	**McaType**. Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 02Eh. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. 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[4...5]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

_instCS0_aliasMSR; MSRC000_2146

_instCS1_aliasMSR; MSRC000_2156

Bits	Description
63:48	Reserved.
47:32	**Syndrome**. Read-write,Volatile. Reset: Cold,0000h. 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	**ErrorPriority**. 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	**Length**. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the

	location of the error. Decoding is available in Table 57 [MCA_SYND_CS Register].

Table 57: MCA_SYND_CS Register

Error Type	Bits	Description
FTI_ILL_REQ	[8:0]	
FTI_ADDR_VIOL	[8:0]	
FTI_SEC_VIOL	[8:0]	
FTI_ILL_RSP	[2:0]	
FTI_RSP_NO_MTCH	[2:0]	
FTI_PAR_ERR	[7:0]	
SDP_PAR_ERR	[7:0]	
ATM_PAR_ERR	[7:0]	
SPF_ECC_ERR	[17:0]	

MSRC000_21[4...5]8 [CS Machine Check Deferred Error Status] (MCA::CS::MCA_DESTAT_CS)

Read-write,Volatile. Reset: Cold,0000_0000_0000_0000h.	
Holds status information for the first deferred error seen in this bank.	
_instCS0_aliasMSR; MSRC000_2148	
_instCS1_aliasMSR; MSRC000_2158	
Bits	**Description**
63	**Val**. Read-write,Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	**Overflow**. 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	Reserved.
58	**AddrV**. Read-write,Volatile. Reset: Cold,0. 1=MCA::CS::MCA_DEADDR_CS contains address information associated with the error.
57:54	Reserved.
53	**SyndV**. 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	Reserved.
44	**Deferred**. 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.
43:0	Reserved.

MSRC000_21[4...5]9 [CS Deferred Error Address] (MCA::CS::MCA_DEADDR_CS)

Read-write,Volatile. 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_DEADDR_CS[LSB].	
_instCS0_aliasMSR; MSRC000_2149	
_instCS1_aliasMSR; MSRC000_2159	
Bits	**Description**
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::CS::MCA_DEADDR_CS[ErrorAddr]. A value of 0 indicates that MCA::CS::MCA_DEADDR_CS[55:0] contains a valid byte address. A value of 6 indicates that MCA::CS::MCA_DEADDR_CS[55:6] contains a valid

	cache line address and that MCA::CS::MCA_DEADDR_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_DEADDR_CS[55:12] contain a valid 4KB memory page and that MCA::CS::MCA_DEADDR_CS[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. 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_DEADDR_CS[LSB].

MSRC001_041[4...5] [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_aliasMSR; MSRC001_0414

_instCS1_aliasMSR; MSRC001_0415

Bits	Description
63:9	Reserved.
8	**SPF_ECC_ERR**. Read-write. Reset: 0. Probe Filter ECC Error: An ECC error occurred on a probe filter access.
7	**ATM_PAR_ERR**. Read-write. Reset: 0. Atomic Request Parity Error: Parity error on read of an atomic transaction.
6	**SDP_PAR_ERR**. Read-write. Reset: 0. Read Response Parity Error: Parity error on incoming read response data.
5	**FTI_PAR_ERR**. Read-write. Reset: 0. Request or Probe Parity Error: Parity error on incoming request or probe response data.
4	**FTI_RSP_NO_MTCH**. Read-write. Reset: 0. Unexpected Response: A response was received from the transport layer which does not match any request.
3	**FTI_ILL_RSP**. Read-write. Reset: 0. Illegal Response: An illegal response was received from the transport layer.
2	**FTI_SEC_VIOL**. Read-write. Reset: 0. Security Violation: A security violation was received from the transport layer.
1	**FTI_ADDR_VIOL**. Read-write. Reset: 0. Address Violation: An address violation was received from the transport layer.
0	**FTI_ILL_REQ**. Read-write. Reset: 0. Illegal Request: An illegal request was received from the transport layer.

3.2.5.9 PIE

MSR0000_0458...MSRC000_2160 [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.

_instPIE_aliasMSRLEGACY; MSR0000_0458

_instPIE_aliasMSR; MSRC000_2160

Bits	Description
63:4	Reserved.
3	**FTI_DAT_STAT**. Read-write. Reset: 0. Poison data consumption: Poison data was written to an internal PIE register.
2	**GMI**. Read-write. Reset: 0. Link Error: An error occurred on a GMI or xGMI link.
1	**CSW**. Read-write. Reset: 0. Register security violation: A security violation was detected on an access to an internal PIE register.
0	**HW_ASSERT**. Read-write. Reset: 0. Hardware Assert: A hardware assert was detected.

MSR0000_0459...MSRC000_2161 [PIE Machine Check Status] (MCA::PIE::MCA_STATUS_PIE)

Reset: Cold,0000_0000_0000_0000h.

Logs information associated with errors.

_instPIE_aliasMSRLEGACY; MSR0000_0459	
_instPIE_aliasMSR; MSRC000_2161	
Bits	**Description**
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::PIE::MCA_MISC0_PIE.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.

37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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	**ErrorCode**. 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 58: MCA_STATUS_PIE[ErrorCodeExt] Decode

Error Type	Bits	Description
HW_ASSERT	[6:0]	0x0
CSW	[6:0]	0x1
GMI	[6:0]	0x2
FTI_DAT_STAT	[6:0]	0x3

MSR0000_045A...MSRC000_2162 [PIE Machine Check Address] (MCA::PIE::MCA_ADDR_PIE)

Read-only. 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.

_instPIE_aliasMSRLEGACY; MSR0000_045A

_instPIE_aliasMSR; MSRC000_2162

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-only. 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.
55:0	**ErrorAddr**. Read-only. Reset: Cold,00_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::PIE::MCA_STATUS_PIE.

Table 59: MCA_ADDR_PIE Register

Error Type	Bits	Description
HW_ASSERT	[55:0]	Reserved
CSW	[55:0]	Reserved
GMI	[55:0]	Reserved
FTI_DAT_STAT	[55:0]	Reserved

MSR0000_045B...MSRC000_2163 [PIE Machine Check Miscellaneous 0] (MCA::PIE::MCA_MISC0_PIE)

Log miscellaneous information associated with errors.

_instPIE_aliasMSRLEGACY; MSR0000_045B

_instPIE_aliasMSR; MSRC000_2163

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.

	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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	**LvtOffset**. 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	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::PIE::MCA_MISC0_PIE[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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	**Ovrflw**. 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	**ErrCnt**. 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	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2164 [PIE Machine Check Configuration] (MCA::PIE::MCA_CONFIG_PIE)

Reset: 0000_0000_0000_0025h.

Controls configuration of the associated machine check bank.

_instPIE_aliasMSR; MSRC000_2164

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**LogDeferredInMcaStat**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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	**McaX**. 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_2165 [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.

_instPIE_aliasMSR; MSRC000_2165

Bits	Description
63:48	**McaType**. Read-only. Reset: 0001h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 02Eh. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

MSRC000_2166 [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

_instPIE_aliasMSR; MSRC000_2166

Bits	Description
63:33	Reserved.
32	**Syndrome**. Read-write,Volatile. Reset: Cold,0. 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	**ErrorPriority**. 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	**Length**. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 60 [MCA_SYND_PIE Register].

Table 60: MCA_SYND_PIE Register

Error Type	Bits	Description
HW_ASSERT	[17:0]	Reserved
CSW	[17:0]	Reserved
GMI	[4:0]	
FTI_DAT_STAT	[3:0]	

MSRC000_2168 [PIE Machine Check Deferred Error Status] (MCA::PIE::MCA_DESTAT_PIE)

Read-write,Volatile. Reset: Cold,0000_0000_0000_0000h.

Holds status information for the first deferred error seen in this bank.

_instPIE_aliasMSR; MSRC000_2168

Bits	Description
63	**Val**. Read-write,Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	**Overflow**. 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	Reserved.
58	**AddrV**. Read-write,Volatile. Reset: Cold,0. 1=MCA::PIE::MCA_DEADDR_PIE contains address information associated with the error.
57:54	Reserved.
53	**SyndV**. 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	Reserved.
44	**Deferred**. 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.
43:0	Reserved.

MSRC000_2169 [PIE Deferred Error Address] (MCA::PIE::MCA_DEADDR_PIE)

Read-write,Volatile. 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_DEADDR_PIE[LSB].

_instPIE_aliasMSR; MSRC000_2169

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::PIE::MCA_DEADDR_PIE[ErrorAddr]. A value of 0 indicates that MCA::PIE::MCA_DEADDR_PIE[55:0] contains a valid byte address. A value of 6 indicates that MCA::PIE::MCA_DEADDR_PIE[55:6] contains a valid cache line address and that MCA::PIE::MCA_DEADDR_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_DEADDR_PIE[55:12] contain a valid 4KB memory page and that MCA::PIE::MCA_DEADDR_PIE[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. 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_DEADDR_PIE[LSB].

MSRC001_0416 [PIE Machine Check Control Mask] (MCA::PIE::MCA_CTL_MASK_PIE)	
Read-write. Reset: 0000_0000_0000_0000h.	
Inhibit detection of an error source.	
_instPIE_aliasMSR; MSRC001_0416	
Bits	**Description**
63:4	Reserved.
3	**FTI_DAT_STAT**. Read-write. Reset: 0. Poison data consumption: Poison data was written to an internal PIE register.
2	**GMI**. Read-write. Reset: 0. Link Error: An error occurred on a GMI or xGMI link.
1	**CSW**. Read-write. Reset: 0. Register security violation: A security violation was detected on an access to an internal PIE register.
0	**HW_ASSERT**. Read-write. Reset: 0. Hardware Assert: A hardware assert was detected.

3.2.5.10 UMC

MSR0000_043C...MSRC000_2100 [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.	
_instUMC_umc0_aliasMSRLEGACY; MSR0000_043C	
_instUMC_umc1_aliasMSRLEGACY; MSR0000_0440	
_instUMC_umc0_aliasMSR; MSRC000_20F0	
_instUMC_umc1_aliasMSR; MSRC000_2100	
Bits	**Description**
63:6	Reserved.
5	**WriteDataCrcErr**. Read-write. Reset: 0. Write data CRC error. A write data CRC error on the DRAM data bus.
4	**AddressCommandParityErr**. Read-write. Reset: 0. Address/command parity error. A parity error on the DRAM address/command bus.
3	**ApbErr**. Read-write. Reset: 0. Advanced peripheral bus error. An error on the advanced peripheral bus.
2	**SdpParityErr**. Read-write. Reset: 0. SDP parity error. A parity error on write data from the data fabric.
1	**WriteDataPoisonErr**. Read-write. Reset: 0. Data poison error.
0	**DramEccErr**. Read-write. Reset: 0. DRAM ECC error. An ECC error on a DRAM read.

MSR0000_043D...MSRC000_2101 [UMC Machine Check Status] (MCA::UMC::MCA_STATUS_UMC)	
Reset: Cold,0000_0000_0000_0000h.	
Logs information associated with errors.	
_instUMC_umc0_aliasMSRLEGACY; MSR0000_043D	
_instUMC_umc1_aliasMSRLEGACY; MSR0000_0441	
_instUMC_umc0_aliasMSR; MSRC000_20F1	
_instUMC_umc1_aliasMSR; MSRC000_2101	
Bits	**Description**
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. 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	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::UMC::MCA_MISC0_UMC.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. 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	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and 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	Reserved.
53	**SyndV**. 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.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.

15:0	**ErrorCode**. 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 61: MCA_STATUS_UMC[ErrorCodeExt] Decode

Error Type	Bits	Description
DramEccErr	[6:0]	0x0
WriteDataPoisonErr	[6:0]	0x1
SdpParityErr	[6:0]	0x2
ApbErr	[6:0]	0x3
AddressCommandParityErr	[6:0]	0x4
WriteDataCrcErr	[6:0]	0x5

MSR0000_043E...MSRC000_2102 [UMC Machine Check Address] (MCA::UMC::MCA_ADDR_UMC)

Read-write,Volatile. 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.

_instUMC_umc0_aliasMSRLEGACY; MSR0000_043E

_instUMC_umc1_aliasMSRLEGACY; MSR0000_0442

_instUMC_umc0_aliasMSR; MSRC000_20F2

_instUMC_umc1_aliasMSR; MSRC000_2102

Bits	Description
63:62	Reserved.
61:56	**LSB**. 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.
55:0	**ErrorAddr**. 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 62: MCA_ADDR_UMC Register

Error Type	Bits	Description
DramEccErr	[55:39] [39:4]	Reserved Reserved
WriteDataPoisonErr	[55:0]	Reserved
SdpParityErr	[55:0]	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]	Reserved Reserved Chip Select

	[31:0]	Reserved

MSR0000_043F...MSRC000_2103 [UMC Machine Check Miscellaneous 0] (MCA::UMC::MCA_MISC0_UMC)

Log miscellaneous information associated with errors.

_instUMC_umc0_aliasMSRLEGACY; MSR0000_043F

_instUMC_umc1_aliasMSRLEGACY; MSR0000_0443

_instUMC_umc0_aliasMSR; MSRC000_20F3

_instUMC_umc1_aliasMSR; MSRC000_2103

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present. AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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	**LvtOffset**. 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	**CntEn**. Reset: 0. 1=Count thresholding errors. AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::UMC::MCA_MISC0_UMC[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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	**Ovrflw**. 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	**ErrCnt**. 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	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2[0F...10]4 [UMC Machine Check Configuration] (MCA::UMC::MCA_CONFIG_UMC)

Reset: 0000_0000_0000_0025h.

Controls configuration of the associated machine check bank.	
_instUMC_umc0_aliasMSR; MSRC000_20F4	
_instUMC_umc1_aliasMSR; MSRC000_2104	
Bits	**Description**
63:39	Reserved.
38:37	**DeferredIntType**. 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	**LogDeferredInMcaStat**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. 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	**DeferredErrorLoggingSupported**. 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	**McaX**. 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_2[0F...10]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.	
_instUMC_umc0_aliasMSR; MSRC000_20F5	
_instUMC_umc1_aliasMSR; MSRC000_2105	
Bits	**Description**
63:48	**McaType**. Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 096h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

MSRC000_2[0F...10]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
_instUMC_umc0_aliasMSR; MSRC000_20F6
_instUMC_umc1_aliasMSR; MSRC000_2106

Bits	Description
63:48	Reserved.
47:32	**Syndrome**. Read-write,Volatile. Reset: Cold,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	**ErrorPriority**. 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	**Length**. 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	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 63 [MCA_SYND_UMC Register].

Table 63: MCA_SYND_UMC Register

Error Type	Bits	Description
DramEccErr	[17:16]	Reserved
	[15]	Software-Managed Bad Symbol ID Error
	[14]	Reserved
	[13:8]	Symbol. Only contains valid information if ErrorPriority indicates a corrected error.
	[7]	Reserved
	[6:4]	Cid. Specifies the rank multiply ID for supported DIMMs.
	[3]	Reserved
	[2:0]	Chip Select
WriteDataPoisonErr	[17:0]	Reserved
SdpParityErr	[17:0]	Reserved
ApbErr	[17:0]	Reserved
AddressCommandParityErr	[17:0]	Reserved
WriteDataCrcErr	[17:0]	Reserved

MSRC000_2[0F...10]8 [UMC Machine Check Deferred Error Status] (MCA::UMC::MCA_DESTAT_UMC)

Read-write,Volatile. Reset: Cold,0000_0000_0000_0000h.

Holds status information for the first deferred error seen in this bank.

_instUMC_umc0_aliasMSR; MSRC000_20F8

_instUMC_umc1_aliasMSR; MSRC000_2108

Bits	Description
63	**Val**. Read-write,Volatile. Reset: Cold,0. 1=A valid error has been detected (whether it is enabled or not).
62	**Overflow**. 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	Reserved.
58	**AddrV**. Read-write,Volatile. Reset: Cold,0. 1=MCA::UMC::MCA_DEADDR_UMC contains address information associated with the error.
57:54	Reserved.
53	**SyndV**. 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	Reserved.
44	**Deferred**. 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.
43:0	Reserved.

MSRC000_2[0F...10]9 [UMC Deferred Error Address] (MCA::UMC::MCA_DEADDR_UMC)

Read-write,Volatile. 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_DEADDR_UMC[LSB].

_instUMC_umc0_aliasMSR; MSRC000_20F9

_instUMC_umc1_aliasMSR; MSRC000_2109

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-write,Volatile. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::UMC::MCA_DEADDR_UMC[ErrorAddr]. A value of 0 indicates that MCA::UMC::MCA_DEADDR_UMC[55:0] contains a valid byte address. A value of 6 indicates that MCA::UMC::MCA_DEADDR_UMC[55:6] contains a valid cache line address and that MCA::UMC::MCA_DEADDR_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_DEADDR_UMC[55:12] contain a valid 4KB memory page and that MCA::UMC::MCA_DEADDR_UMC[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. 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_DEADDR_UMC[LSB].

MSRC000_2[0F...10]A [UMC Machine Check Miscellaneous 1] (MCA::UMC::MCA_MISC1_UMC)

Read-write.

Log miscellaneous information associated with errors, as defined by each error type.

_instUMC_umc0_aliasMSR; MSRC000_20FA

_instUMC_umc1_aliasMSR; MSRC000_210A

Bits	Description
63	**Valid**. Read-write. Reset: 1. 1=A valid CntP field is present in this register.
62	**CntP**. Read-write. Reset: 1. 1=A valid threshold counter is present.
61	**Locked**. Read-write. 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.
60	**IntP**. Read-write. Reset: 1. 1=ThresholdIntType can be used to generate interrupts. 0=ThresholdIntType and interrupt generation are not supported.
59:52	Reserved.
51	**CntEn**. Read-write. Reset: 0. 1=Count thresholding errors.
50:49	**ThresholdIntType**. Read-write. Reset: Cold,0h. Specifies the type of interrupt signaled when Ovrflw is set. 00b = No interrupt. 01b = APIC based interrupt (see Core::X86::Msr::McaIntrCfg[ThresholdLvtOffset]) to all cores. 10b = SMI trigger event. 11b = Reserved.
48	**Ovrflw**. Read-write. 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.
47:44	Reserved.
43:32	**ErrCnt**. Read-write. 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.
31:24	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC001_04[0F...10] [UMC Machine Check Control Mask] (MCA::UMC::MCA_CTL_MASK_UMC)

Read-write. Reset: 0000_0000_0000_0000h.

Inhibit detection of an error source.

_instUMC_umc0_aliasMSR; MSRC001_040F

_instUMC_umc1_aliasMSR; MSRC001_0410

Bits	Description
63:6	Reserved.
5	**WriteDataCrcErr**. Read-write. Reset: 0. Write data CRC error. A write data CRC error on the DRAM data bus.
4	**AddressCommandParityErr**. Read-write. Reset: 0. Address/command parity error. A parity error on the DRAM address/command bus.
3	**ApbErr**. Read-write. Reset: 0. Advanced peripheral bus error. An error on the advanced peripheral bus.
2	**SdpParityErr**. Read-write. Reset: 0. SDP parity error. A parity error on write data from the data fabric.
1	**WriteDataPoisonErr**. Read-write. Reset: 0. Data poison error.
0	**DramEccErr**. Read-write. Reset: 0. DRAM ECC error. An ECC error on a DRAM read.

3.2.5.11 Parameter Block

MSR0000_044C...MSRC000_2130 [PB Machine Check Control] (MCA::PB::MCA_CTL_PB)

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::PB::MCA_CTL_PB register must be enabled by the corresponding enable bit in Core::X86::Msr::MCG_CTL. Does not affect error detection, correction, or logging.

_instPB_aliasMSRLEGACY; MSR0000_044C

_instPB_aliasMSR; MSRC000_2130

Bits	Description
63:1	Reserved.
0	**EccError**. Read-write. Reset: 0. An ECC error in the Parameter Block RAM array.

MSR0000_044D...MSRC000_2131 [PB Machine Check Status] (MCA::PB::MCA_STATUS_PB)

Reset: Cold,0000_0000_0000_0000h.

Logs information associated with errors.

_instPB_aliasMSRLEGACY; MSR0000_044D

_instPB_aliasMSR; MSRC000_2131

Bits	Description
63	**Val**. 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	**Overflow**. 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	**UC**. 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	**En**. Reset: Cold,0. 1=MCA error reporting is enabled for this error, as indicated by the corresponding bit in MCA::PB::MCA_CTL_PB. This bit is a copy of bit in MCA::PB::MCA_CTL_PB for this error.

	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
59	**MiscV**. Reset: Cold,0. 1=Valid thresholding in MCA::PB::MCA_MISC0_PB.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
58	**AddrV**. Reset: Cold,0. 1=MCA::PB::MCA_ADDR_PB contains address information associated with the error.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
57	**PCC**. 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	**ErrCoreIdVal**. 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	**TCC**. Reset: Cold,0. 0=The thread which consumed the error can be restarted reliably at the instruction pointer address pushed onto the exception handler stack if any uncorrected error has been corrected by software. 1=The thread which consumed the error is not restartable and must be terminated. Only meaningful when MCA::PB::MCA_STATUS_PB[PCC]=0.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
54	Reserved.
53	**SyndV**. Reset: Cold,0. 1=This error logged information in MCA::PB::MCA_SYND_PB. If MCA::PB::MCA_SYND_PB[ErrorPriority] is the same as the priority of the error in MCA::PB::MCA_STATUS_PB, then the information in MCA::PB::MCA_SYND_PB is associated with the error in MCA::PB::MCA_STATUS_PB.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read,Write-0-only,Error-on-write-1.
52:47	Reserved.
46	**CECC**. 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	**UECC**. 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	**Deferred**. 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	**Poison**. 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.
42:38	Reserved.
37:32	**ErrCoreId**. 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.
31:22	Reserved.
21:16	**ErrorCodeExt**. 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::PB::MCA_CTL_PB 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	**ErrorCode**. 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 64: MCA_STATUS_PB[ErrorCodeExt] Decode

Error Type	Bits	Description
EccError	[6:0]	0x0

MSR0000_044E...MSRC000_2132 [PB Machine Check Address] (MCA::PB::MCA_ADDR_PB)

Read-only. Reset: Cold,0000_0000_0000_0000h.

MCA::PB::MCA_ADDR_PB stores an address and other information associated with the error in MCA::PB::MCA_STATUS_PB. The register is only meaningful if MCA::PB::MCA_STATUS_PB[Val]=1 and MCA::PB::MCA_STATUS_PB[AddrV]=1.

_instPB_aliasMSRLEGACY; MSR0000_044E

_instPB_aliasMSR; MSRC000_2132

Bits	Description
63:62	Reserved.
61:56	**LSB**. Read-only. Reset: Cold,00h. Specifies the least significant valid bit of the address contained in MCA::PB::MCA_ADDR_PB[ErrorAddr]. A value of 0 indicates that MCA::PB::MCA_ADDR_PB[55:0] contains a valid byte address. A value of 6 indicates that MCA::PB::MCA_ADDR_PB[55:6] contains a valid cache line address and that MCA::PB::MCA_ADDR_PB[5:0] are not part of the address and should be ignored by error handling software. A value of 12 indicates that MCA::PB::MCA_ADDR_PB[55:12] contain a valid 4KB memory page and that MCA::PB::MCA_ADDR_PB[11:0] should be ignored by error handling software.
55:0	**ErrorAddr**. Read-only. Reset: Cold,00_0000_0000_0000h. Contains the address, if any, associated with the error logged in MCA::PB::MCA_STATUS_PB.

Table 65: MCA_ADDR_PB Register

Error Type	Bits	Description
EccError	[55:0]	Reserved

MSR0000_044F...MSRC000_2133 [PB Machine Check Miscellaneous 0] (MCA::PB::MCA_MISC0_PB)

Log miscellaneous information associated with errors.

_instPB_aliasMSRLEGACY; MSR0000_044F

_instPB_aliasMSR; MSRC000_2133

Bits	Description
63	**Valid**. Reset: 1. 1=A valid CntP field is present in this register.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
62	**CntP**. Reset: 1. 1=A valid threshold counter is present.
	AccessType: Core::X86::Msr::HWCR[McStatusWrEn] ? Read-write : Read-only.
61	**Locked**. 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	**IntP**. 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::PB::MCA_MISC0_PB[Locked]) ? Read-write : Read-only.
59:56	Reserved.
55:52	**LvtOffset**. 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::PB::MCA_MISC0_PB[Locked]) ? Read-write : Read-only.
51	**CntEn**. Reset: 0. 1=Count thresholding errors.
	AccessType: (Core::X86::Msr::HWCR[McStatusWrEn] | !MCA::PB::MCA_MISC0_PB[Locked]) ? Read-write : Read-only.
50:49	**ThresholdIntType**. 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::PB::MCA_MISC0_PB[Locked]) ? Read-write : Read-only.
48	**Ovrflw**. 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::PB::MCA_MISC0_PB[Locked]) ? Read-write : Read-only.
47:44	Reserved.
43:32	**ErrCnt**. 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::PB::MCA_MISC0_PB[Locked]) ? Read-write : Read-only.
31:24	**BlkPtr**. Read-write. Reset: 01h. 00h=Extended MISC MSR block is not valid. 01h=Extended MSR block is valid.
23:0	Reserved.

MSRC000_2134 [PB Machine Check Configuration] (MCA::PB::MCA_CONFIG_PB)

Reset: 0000_0000_0000_0021h.

Controls configuration of the associated machine check bank.

_instPB_aliasMSR; MSRC000_2134

Bits	Description
63:39	Reserved.
38:37	**DeferredIntType**. 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	**McaXEnable**. 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.
31:6	Reserved.
5	**DeferredIntTypeSupported**. Read-only. Reset: 1. 1=MCA::PB::MCA_CONFIG_PB[DeferredIntType] controls the type of interrupt generated on a deferred error. Deferred errors are supported in this bank only if MCA::PB::MCA_CONFIG_PB[DeferredErrorLoggingSupported]=1.
4:3	Reserved.
2	**DeferredErrorLoggingSupported**. 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. MCA_DESTAT and MCA_DEADDR are supported in this MCA bank. 0=Deferred errors are not supported in this bank.
1	Reserved.
0	**McaX**. Read-only. Reset: 1. 1=This bank provides Machine Check Architecture Extensions. Up to 4 additional MISC registers (MISC1-MISC4) are supported. MCA::PB::MCA_MISC0_PB[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::PB::MCA_STATUS_PB[TCC] is present.

MSRC000_2135 [PB IP Identification] (MCA::PB::MCA_IPID_PB)

Reset: 0000_0005_0000_0000h.

The MCA::PB::MCA_IPID_PB register is used by software to determine what IP type and revision is associated with the MCA bank.

_instPB_aliasMSR; MSRC000_2135

Bits	Description
63:48	**McaType**. Read-only. Reset: 0000h. The McaType of the MCA bank within this IP.
47:44	Reserved.
43:32	**HardwareID**. Read-only. Reset: 005h. The Hardware ID of the IP associated with this MCA bank.
31:0	**InstanceId**. Read-write. Reset: 0000_0000h. The instance ID of this IP. This is initialized to a unique ID per instance of this register.

MSRC000_2136 [PB Machine Check Syndrome] (MCA::PB::MCA_SYND_PB)

Read-write,Volatile. Reset: Cold,0000_0000_0000_0000h.

Logs physical location information associated with error in MCA::PB::MCA_STATUS_PB Thread 0

_instPB_aliasMSR; MSRC000_2136

Bits	Description
63:33	Reserved.
32	**Syndrome**. Read-write,Volatile. Reset: Cold,0. Contains the syndrome, if any, associated with the error logged in MCA::PB::MCA_STATUS_PB. The low-order bit of the syndrome is stored in bit 0, and the syndrome has a length specified by MCA::PB::MCA_SYND_PB[Length]. The Syndrome field is only valid when MCA::PB::MCA_SYND_PB[Length] is not 0.
31:27	Reserved.
26:24	**ErrorPriority**. Read-write,Volatile. Reset: Cold,0h. Encodes the priority of the error logged in MCA::PB::MCA_SYND_PB. 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	**Length**. Read-write,Volatile. Reset: Cold,00h. Specifies the length in bits of the syndrome contained in MCA::PB::MCA_SYND_PB[Syndrome]. A value of 0 indicates that there is no valid syndrome in MCA::PB::MCA_SYND_PB. For example, a syndrome length of 9 means that MCA::PB::MCA_SYND_PB[Syndrome] bits [8:0] contains a valid syndrome.
17:0	**ErrorInformation**. Read-write,Volatile. Reset: Cold,0_0000h. Contains error-specific information about the location of the error. Decoding is available in Table 66 [MCA_SYND_PB Register].

Table 66: MCA_SYND_PB Register

Error Type	Bits	Description
EccError	[17:0]	Reserved

MSRC001_0413 [PB Machine Check Control Mask] (MCA::PB::MCA_CTL_MASK_PB)

Read-write. Reset: 0000_0000_0000_0000h.

Inhibit detection of an error source.

_instPB_aliasMSR; MSRC001_0413

Bits	Description
63:1	Reserved.
0	**EccError**. Read-write. Reset: 0. An ECC error in the Parameter Block RAM array.

4 System Management Unit (SMU)

4.1 SMU Registers

The system management unit (SMU) is a subcomponent of the processor that is responsible for a variety of system and power management tasks during boot and runtime.

4.1.1 MP Configuration Unit Registers

SEVx00010580 [SEV CmdResp mailbox register.] (MP::MP0CRU::SEVCmdResp)	
Read-write. Reset: 0000_0000h.	
SEVx00010580; SEV=NBIFEPFNCFG::BASE_ADDR_3_instNBIF0_func2_aliasHOST[BASE_ADDR]	
Bits	**Description**
31:0	**CmdMailbox**. Read-write. Reset: 0000_0000h. SEV Cmd Response mailbox register. See: AMD SEV API Specification (PID 55766) for details.

SEVx000105E0 [SEV CmdBufAddr_Lo mailbox register.] (MP::MP0CRU::SEVCmdBufLo)	
Read-write. Reset: 0000_0000h.	
SEVx000105E0; SEV=NBIFEPFNCFG::BASE_ADDR_3_instNBIF0_func2_aliasHOST[BASE_ADDR]	
Bits	**Description**
31:0	**RespMailboxLo**. Read-write. Reset: 0000_0000h. SEV Command buffer address low mailbox register. See: AMD SEV API Specification (PID 55766) for details.

SEVx000105E4 [SEV CmdBufAddr_Hi mailbox register.] (MP::MP0CRU::SEVCmdBufHi)	
Read-write. Reset: 0000_0000h.	
SEVx000105E4; SEV=NBIFEPFNCFG::BASE_ADDR_3_instNBIF0_func2_aliasHOST[BASE_ADDR]	
Bits	**Description**
31:0	**RespMailboxHi**. Read-write. Reset: 0000_0000h. SEV Command buffer address high mailbox register. See: AMD SEV API Specification (PID 55766) for details.

4.2 Thermal (THM)

The thermal block contains all the features related to temperature sensing, control, and reporting. It includes:

- Temperature collection and calculation using TCON (digital control logic) and TMON and Remote Diode Interface macros.
- Fan speed control for off-chip fans.
- Temperature reporting through the SMBUS interface.

Table 67: List of Acronyms and Terms used in Thermal (THM)

Term	Definition
TMON	Thermal Monitor

4.2.1 Registers

SMUTHMx00000000 (SMU::THM::THM_TCON_CUR_TMP)
Reset: 0000_0000h.
Provides the current control temperature (T ctl) after the slew-rate controls have been applied.

_aliasSMN; SMUTHMx00000000; SMUTHM=0005_9800h

Bits	Description
31:21	**CUR_TEMP**. Reset: 000h. Provides current control temperature.
	AccessType: (SMU::THM::THM_TCON_CUR_TMP[CUR_TEMP_TJ_SEL] == 3) ? Read-write : Read-only.
20	Reserved.
19	**CUR_TEMP_RANGE_SEL**. Reset: 0. 0=Report on 0C to 225C scale range. 1=Report on -49C to 206C scale range.
	AccessType: (SMU::THM::THM_TCON_CUR_TMP[CUR_TEMP_TJ_SEL] == 3) ? Read-write : Read-only.
18:0	Reserved.

5 Advanced Platform Management Link (APML)

5.1 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 SBI Temperature Sensor Interface (SB-TSI). For related specifications, see 1.2 [Reference Documents].

5.1.1 Definitions

Table 68: APML Definitions

Term	Description
ARA	Alert response address.
ARP	Address Resolution Protocol
EC	Embedded Controller.
KBC	Keyboard Controller.
Master or SMBus Master	The device that initiates and terminates all communication and drives the clock, SCL.
PEC	Packet error code.
POR	Power on reset.
RTS	Remote temperature sensor, typical examples are ADM1032, LM99, MAX6657, EMC1002.
SBI	Sideband interface.
Slave or SMBus slave	The slave cannot initiate SMBus communication and cannot drive the clock but can drive the data signal SDA and the alert signal ALERT_L.
TSI	Temperature sensor interface.

5.2 SBI Bus Characteristics

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

5.2.1 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.

5.2.2 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.

5.3 SBI Processor Information

5.3.1 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.

5.3.1.1 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.

5.3.2 Processor States

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

5.4 SBI Protocols

5.4.1 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. 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.

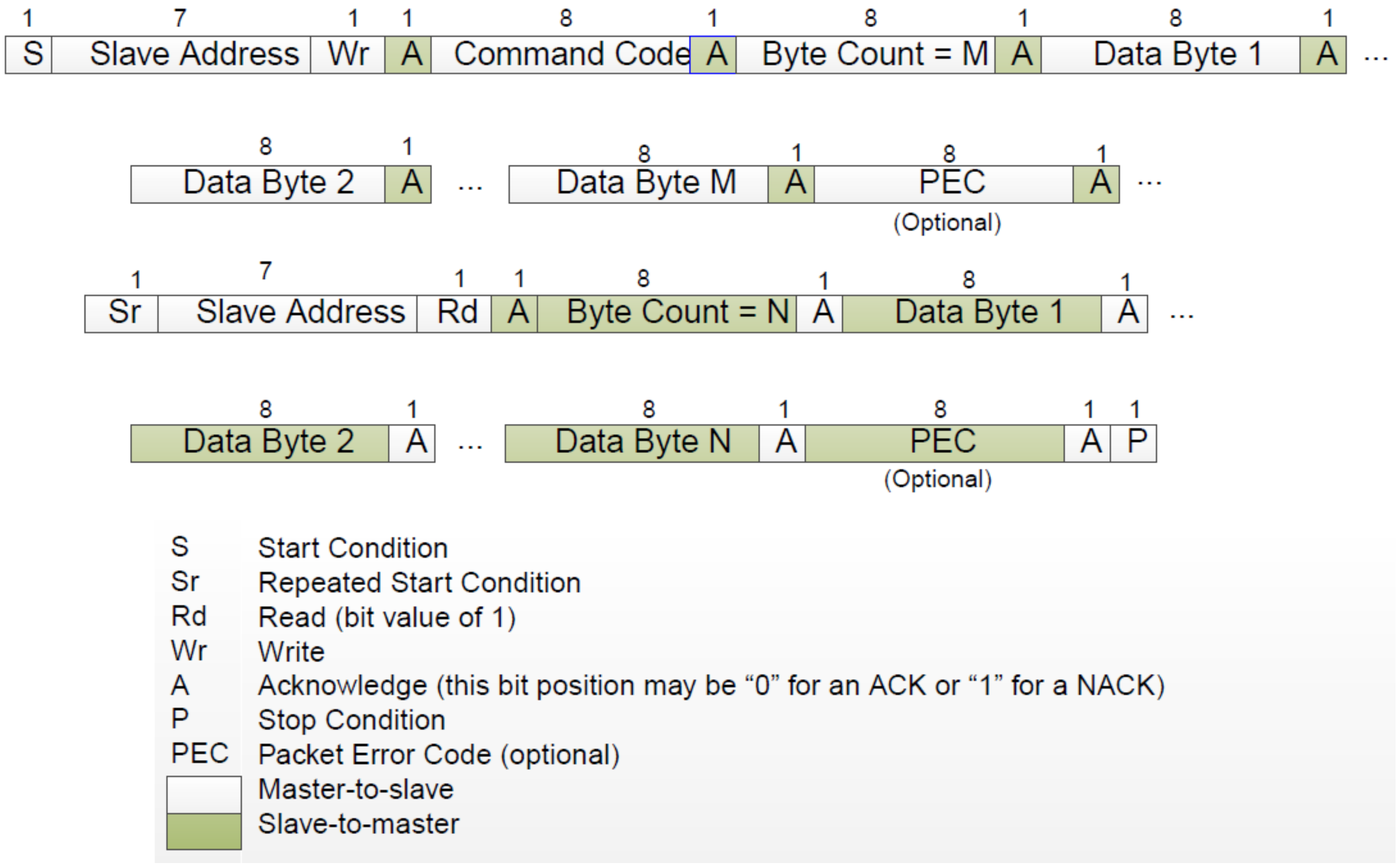


Figure 19: SBI Transmission Protocol

5.4.2 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.

5.4.2.1 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.

5.4.2.1.1 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.

5.4.2.1.2 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.

5.4.2.2 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.

5.4.2.2.1 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 section 3.1.1.3 of the SMBUS Management Guide). 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.

5.5 SBI Physical Interface

5.5.1 SBI SMBus Address

The SMBus address is really 7 bits. Some vendors and the SMBus specification show the address as 8 bits, left justified with the R/W bit as a write (0) making bit 0. Some vendors use only the 7 bits to describe the address.

5.5.2 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.

5.5.3 Pass-FET Option

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

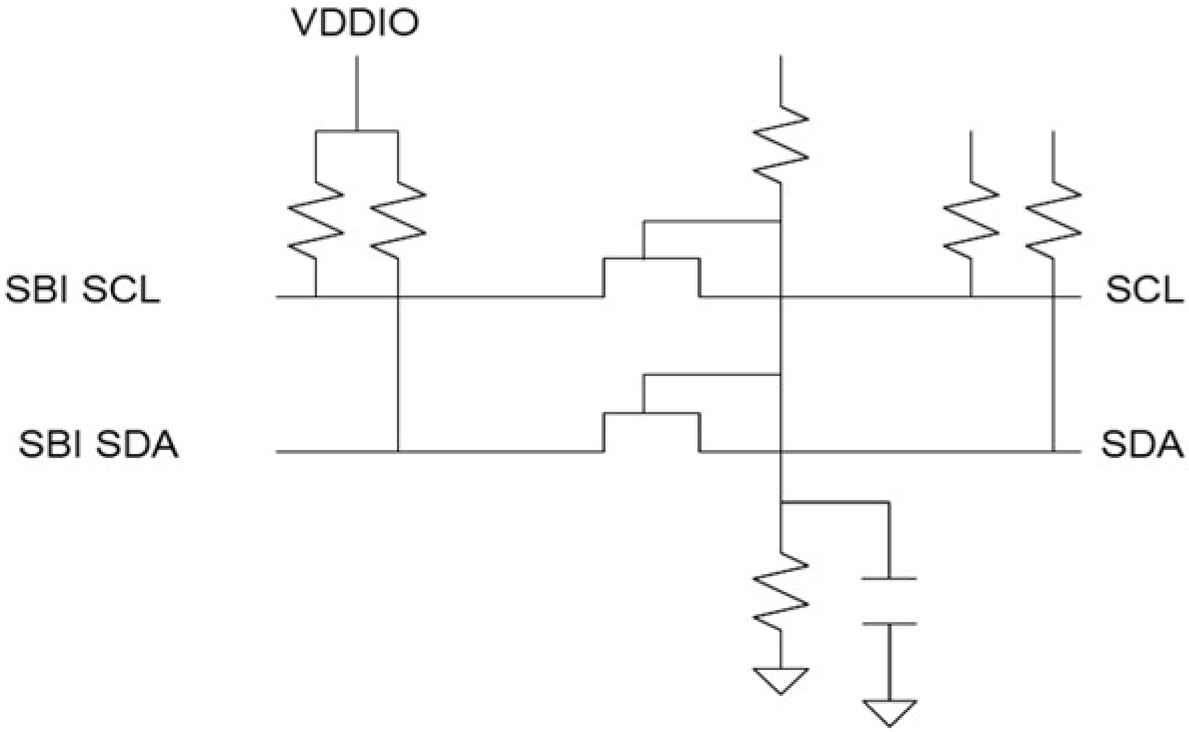


Figure 20: Pass FET Implementation

Notes:

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

6 SB Temperature Sensor Interface (SB-TSI)

6.1 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 21 [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 22 [SB-TSI Thermal Management Example]. SB-TSI supports the SMBus protocols that typical RTS supports.

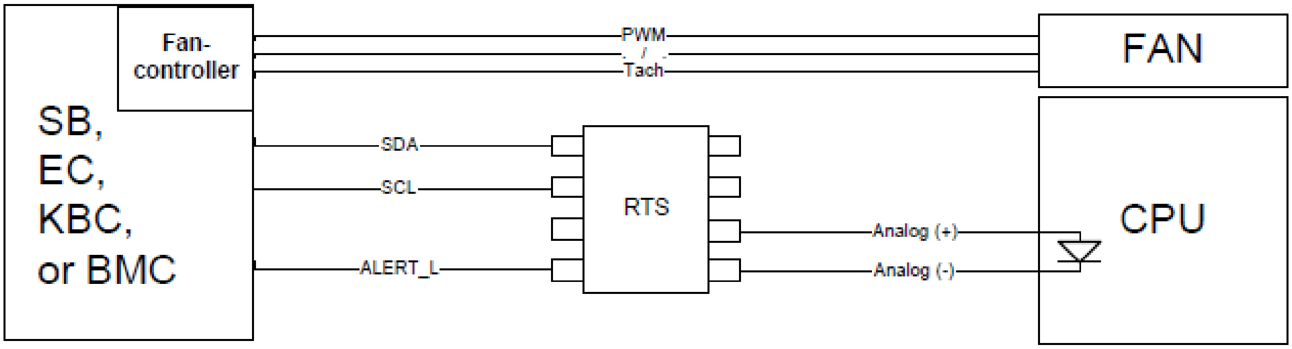


Figure 21: RTS Thermal Management Example

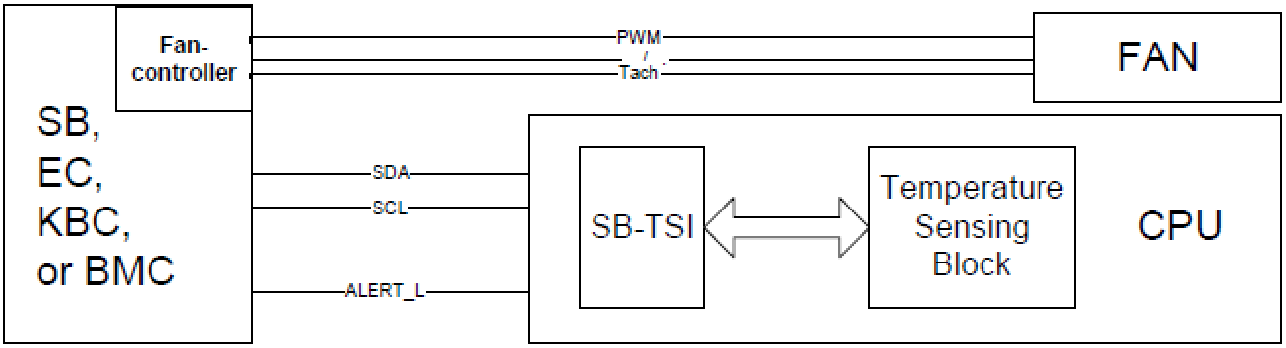


Figure 22: SB-TSI Thermal Management Example

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.

6.1.1 Definitions

Table 69: SB-TSI Definitions

Term	Description
BMC	Base management controller.
TSM	Temperature sensor macro.
SB-TSI	Sideband Internal Temperature Sensor Interface. See APML.

6.2 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 in 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 by written 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.

6.2.1 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.

6.2.1.1 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.

6.2.2 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.

6.2.3 Alert Behavior

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

- SBTSI::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.

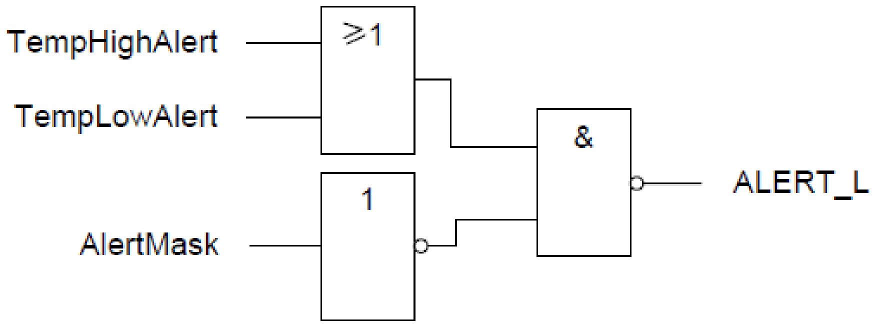


Figure 23: Alert Assertion Diagram

6.2.4 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.

[The SB-TSI Configuration Register] SBTSI::Config[ReadOrder] specifies the order for reading integer and decimal part of the CPU temperature value for atomic CPU temperature reads. If SBTSI::Config[ReadOrder] is 0, then a read of the integer part (SBTSI::CpuTempInt) of the CPU temperature triggers a latch of the decimal part (SBTSI::CpuTempDec) 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].

6.2.5 SB-TSI Temperature and Threshold Encodings

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

Table 70: SB-TSI CPU Temperature and Threshold Encoding Examples

Temperature	Temperature High Byte SBTSI::CpuTempInt[CpuTempInt] SBTSI::HiTempInt[HiTempInt]	Temperature Low Byte SBTSI::CpuTempDec[CpuTempDec] SBTSI::HiTempDec[HiTempDec]

	SBTSI::LoTempInt[LoTempInt]	SBTSI::LoTempDec[LoTempDec]
0.000 °C	0000_0000b	0000_0000b
1.000 °C	0000_0001b	0000_0000b
25.125 °C	0001_1001b	0010_0000b
50.875 °C	0011_0010b	1110_0000b
90.000 °C	0101_1010b	0000_0000b

6.2.6 SB-TSI Temperature Offset Encoding

By default, SBTSI::CpuTempInt and SBTSI::CpuTempDec provide Tctl from the processor. Refer to the BIOS and Kernel Developer's Guide for the processor family for further details on Tctl. The temperature offset registers allow the system to adjust the SB-TSI temperature from Tctl.

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

Table 71: SB-TSI Temperature Offset Encoding Examples

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

6.3 SB-TSI Physical Interface

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

6.3.1 SB-TSI SMBus Address

The SMBus address is really 7 bits. Some vendors and the SMBus specification show the address as 8 bits, left justified with the R/W bit as a write (0) making bit 0. Some vendors use only the 7 bits to describe the address. The addresses can vary with address select pins.

Table 72: 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.

6.3.2 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.

6.3.3 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.

6.3.4 Pass-FET Option

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

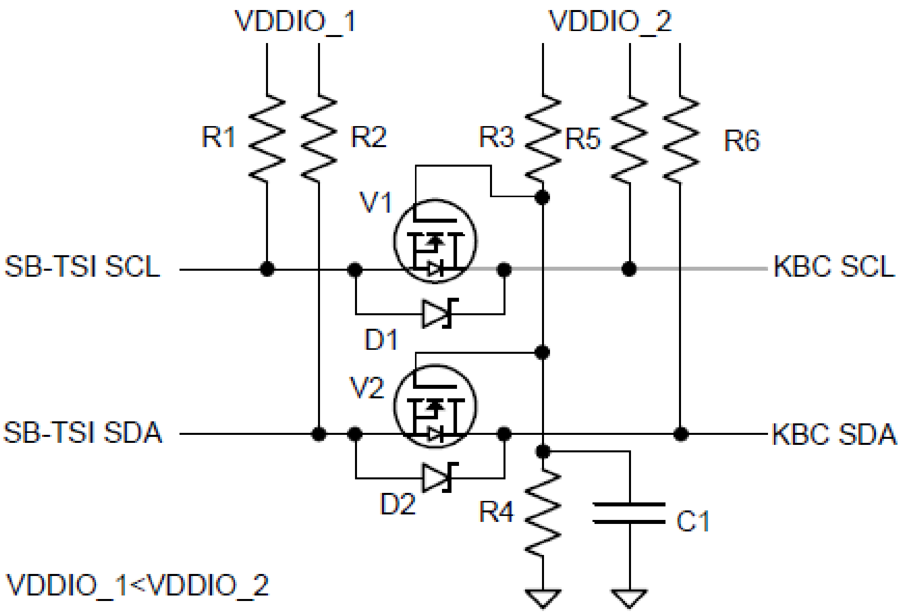


Figure 24: Pass FET Implementation

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 Vgs 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 Vol < Vil on low side)

6.4 SB-TSI Registers

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

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

	portion of the CPU temperature.

SBTSIx02 [SB-TSI Status] (SBTSI::Status)

Read-only,Volatile.

If SBTSI::AlertConfig[AlertCompEn]==0 , the temperature alert is latched high until the alert is read. If SBTSI::AlertConfig[AlertCompEn]==1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. See 6.2.3 [Alert Behavior].

Bits	Description
7:5	Reserved.
4	**TempHighAlert**: **temperature high alert**. Read-only,Volatile. Reset: Cold,X. 1=Indicates that the CPU temperature is greater than or equal to the high temperature threshold (SBTSI::HiTempInt, SBTSI::HiTempDec) for SBTSI::AlertThreshold[AlertThr] consecutive samples. 0=Indicates that the CPU tempera- ture is less than the high temperature threshold (SBTSI::HiTempInt, SBTSI::HiTempDec) for SBTSI::AlertThreshold[AlertThr] samples and SBTSI::AlertConfig[AlertCompEn]=1. Hardware will clear this bit when read if SBTSI::AlertConfig[AlertCompEn]=0.
3	**TempLowAlert**: **temperature low alert**. Read-only,Volatile. Reset: Cold,X. 1=Indicates that the CPU temperature is less than or equal to the low temperature threshold (SBTSI::LoTempInt, SBTSI::LoTempDec) for SBTSI::AlertThreshold[AlertThr] consecutive samples. 0=Indicates the CPU temperature is greater than the low temperature threshold (SBTSI::LoTempInt, SBTSI::LoTempDec) for SBTSI::AlertThreshold[AlertThr] samples and SBTSI::AlertConfig[AlertCompEn]=1. Hardware will clear this bit when read if SBTSI::AlertConfig[AlertCompEn]=0.
2:0	Reserved.

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

Reset: Cold,00h.

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

Bits	Description
7	**AlertMask**: **alert mask**. Read-only,Volatile. Reset: Cold,0. 0=ALERT_L pin enabled. 1=ALERT_L pin disabled and does not assert. 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	**RunStop**: **run stop**. Read-only. Reset: Cold,0. 0=Updates to SBTSI::CpuTempInt and SBTSI::CpuTempDec and the alert comparisons are enabled; Alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]) continue to update. 1=Updates to SBTSI::CpuTempInt and SBTSI::CpuTempDec and the alert comparisons are disabled; Alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]) are stopped. See 6.2.4 [Atomic Read Mechanism] for further details.
5	**ReadOrder**: **atomic read order**. Read-only. Reset: Cold,0. 0=Reading SBTSI::CpuTempInt causes the state of SBTSI::CpuTempDec to be latched. 1=Reading SBTSI::CpuTempDec causes the state of SBTSI::CpuTempInt to be latched. See 6.2.4 [Atomic Read Mechanism] for further details.
4:2	Reserved.
1	**AraDis**: **ARA disable**. 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	**UpRate**: **update rate**. Read-write. Reset: Cold,08h. This field specifies the rate at which CPU temperature is compared against the temperature thresholds to determine if an alert event has occurred. Write access causes a reset of the alert history counters (specified by SBTSI::AlertThreshold[AlertThr]) and the corresponding timer (specified by SBTSI::UpdateRate[UpRate]).

ValidValues:

Value	Description
00h	0.0625 Hz
01h	0.125 Hz
02h	0.25 Hz
03h	0.5 Hz
04h	1 Hz
05h	2 Hz
06h	4 Hz
07h	8 Hz
08h	16 Hz
09h	32 Hz
0Ah	64 Hz
FFh-0Bh	Reserved.

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

Read-write. Reset: Cold,46h.

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

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

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

Read-write. Reset: Cold,00h.

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

Bits	Description
7	**LoTempInt**: **low temperature integer threshold**. Read-write. Reset: Cold,0. This field specifies the integer portion of the low temperature threshold.
6:0	Reserved.

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

Read-write. Reset: Cold,00h.

This register provides write access to SBTSI::Config.

Bits	Description
7	**AlertMask**: **alert mask**. Read-write. Reset: Cold,0. See SBTSI::Config[AlertMask].
6	**RunStop**: **run stop**. Read-write. Reset: Cold,0. See SBTSI::Config[RunStop].
5	**ReadOrder**: **atomic read order**. Read-write. Reset: Cold,0. See SBTSI::Config[ReadOrder].
4:2	Reserved.
1	**AraDis**: **ARA disable**. 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	**CpuTempDec**: **decimal CPU temperature value**. Read-only. Reset: Cold,XXXb. Read-only. This field returns the decimal portion of the CPU temperature.
4:0	Reserved.

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

Read-write. Reset: Cold,00h.

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

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

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

Read-write. Reset: Cold,00h.

See SBTSI::CpuTempOffInt.

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

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

Read-write. Reset: Cold,00h.

See SBTSI::HiTempInt.

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

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

Read-write. Reset: Cold,00h.

See SBTSI::LoTempInt.

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

SBTSIx22 [Timeout Configuration] (SBTSI::TimeoutConfig)

Read-write. Reset: Cold,80h.

Bits	Description
7	**TimeoutEn**: **SMBus timeout enable**. 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)

<table>
<tr><td colspan="2">Read-write. Reset: Cold,00h.</td></tr>
<tr><td colspan="2">See 6.2.3 [Alert Behavior].</td></tr>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:3</td><td>Reserved.</td></tr>
<tr><td rowspan="2">2:0</td><td><b>AlertThr</b>: <b>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.</td></tr>
<tr><td><b>ValidValues</b>:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>1 Sample</td></tr>
<tr><td>6h-1h</td><td>&lt;Value+1> Samples</td></tr>
<tr><td>7h</td><td>8 Samples</td></tr>
</table></td></tr>
</table>

SBTSIxBF [Alert Configuration] (SBTSI::AlertConfig)

Read-write.	
Bits	**Description**
7:1	Reserved.
0	**AlertCompEn**: **alert comparator mode enable**. Read-write. Reset: Cold,X. 0=SBTSI::Status[TempHighAlert] and SBTSI::Status[TempLowAlert] are read 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	**ManId**: **Manufacture ID**. Read-only. Reset: Cold,0. Returns the AMD manufacture ID.

SBTSIxFF [Revision] (SBTSI::Revision)

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

7 UMC

7.1 UMC Overview

Each processor die includes two Unified Memory Controllers (UMC).

Each UMC controls one DDR4 channel, either 64-bit or 72-bit with ECC.

The UMC interfaces to the system via the Data Fabric (DF) Coherent Slave. The UMC operates on normalized addresses which only include address bits within a UMC's populated memory address range. The physical to normalized address translation varies based on various Data Fabric settings. The normalized address to DRAM address protocol translation is performed by the UMC.

The following restrictions limit the DRAM and/or DIMM types and configurations supported by the UMC:

- DDR4 DRAM devices.
- Mixing of DDR4 ECC and non-ECC DIMMs within a channel is not supported.
- Mixing of base module types within a channel is not supported.
- All UMCs are required to operate at the same MEMCLK frequency, regardless of the channel.
- Asymmetrical DRAM package types are not supported.
- Additional AM4 package rules:
    - DDR4 x8 and x16 devices are supported. x4 devices are not supported.
- Additional SP3 and SP4 package rules:
    - DIMMs must be of the same type per die (channel pair).
        - NVDIMM-N DIMMs may be populated with a DIMM of the same DIMM base module type.
    - When populating channel pairs, the following conditions apply:
        - Two DIMMs: Populate one per channel and the DIMMs must be identical.
        - Three DIMMs: The channel with two DIMMs must be identical and have the same total capacity as the other channel with one DIMM.
            - Three NVDIMM-N DIMMs across the two channels is not supported.
        - Four DIMMs: DIMMs must be pair-wise identical.
    - Mixing of x4 and x8 devices within a channel is not supported.
    - Mixing of 3DS and non-3DS devices within a channel is not supported.

7.1.1 UMC Frequency Support

The motherboard should comply with the relevant AMD socket motherboard design guidelines (MBDG) to achieve the rated speeds. In cases where MBDG design options exist, lower-quality options may compromise the maximum achievable speed; motherboard designers should assess the tradeoffs.

List of Namespaces

Namespace	Heading(s)
Core::X86::Apic	2.1.12.2.2 [Local APIC Registers]
Core::X86::Cpuid	2.1.13.1 [CPUID Instruction Functions]
Core::X86::Msr	2.1.14.1 [MSRs - MSR0000_xxxx] 2.1.14.2 [MSRs - MSRC000_0xxx] 2.1.14.3 [MSRs - MSRC001_0xxx] 2.1.14.4 [MSRs - MSRC001_1xxx]
Core::X86::Pmc::Core	2.1.15.3 [Large Increment per Cycle Events] 2.1.15.4.1 [Floating Point (FP) Events] 2.1.15.4.2 [LS Events] 2.1.15.4.3 [IC and BP Events] 2.1.15.4.4 [DE Events] 2.1.15.4.5 [EX (SC) Events] 2.1.15.4.6 [L2 Cache Events.]
Core::X86::Pmc::L3	2.1.15.5.1 [L3 Cache PMC Events]
Core::X86::Smm	2.1.12.1.6 [System Management State]
IO	2.1.8 [PCI Configuration Legacy Access]
MCA::CS	3.2.5.8 [CS]
MCA::DE	3.2.5.4 [DE]
MCA::EX	3.2.5.5 [EX]
MCA::FP	3.2.5.6 [FP]
MCA::IF	3.2.5.2 [IF]
MCA::L2	3.2.5.3 [L2]
MCA::L3	3.2.5.7 [L3 Cache]
MCA::LS	3.2.5.1 [LS]
MCA::PB	3.2.5.11 [Parameter Block]
MCA::PIE	3.2.5.9 [PIE]
MCA::UMC	3.2.5.10 [UMC]
MP::MP0CRU	4.1.1 [MP Configuration Unit Registers]
SBTSI	6.4 [SB-TSI Registers]
SMU::THM	4.2.1 [Registers]

List of Definitions

ABS: ABS(integer expression): Remove sign from signed value.

AGESA™: AMD Generic Encapsulated Software Architecture.

AM4: Desktop, single die, single socket. For client platform DDR4. AM4 = (Core::X86::Cpuid::BrandId[PkgType] == 02h).

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.

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.

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.

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[7:0]]/Core::X86::Msr::PStateDef[CpuDfsId])*200.

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://www.acpi.info.

docAM4: Socket AM4 Processor Functional Data Sheet, order# 55509.

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.

docI2C: I2C Bus Specification. 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.

docPCIlb: PCI Local Bus Specification. http://www.pcisig.org.

docRAS: RAS Feature Enablement for AMD Family 17h Models 00h-0Fh, order# 55987.

docRevG: Revision Guide for AMD Family 17h Models 00h-0Fh Processors, order# 55449.

Doubleword: A 32-bit value.

DW: Doubleword.

EC: Embedded Controller.

ECS: Extended Configuration Space.

EDC: Electrical design current. Indicates the maximum current the voltage rail can demand for a short, thermally insignificant time.

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.

FDS: Functional Data Sheet. There is one FDS for each package type. See docSAM4.

FID: Frequency Identifier. Specifies the PLL frequency multiplier for a given clock domain.

FLOOR: FLOOR(integer expression): Rounds real number down to nearest integer.

GB: Gbyte or Gigabyte; 1,073,741,824 bytes.

HTC: Hardware Thermal Control.

IBS: Instruction based sampling.

Inaccessible: Not readable or writable (e.g., Hide ? Inaccessible : Read-Write).

IORR: IO range register.

KB: Kbyte or Kilobyte; 1024 bytes.

KBC: Keyboard Controller.

L3: Level 3 Cache. The L3 term is also in Addrmaps to enumerate CCX units.

LINT: Local interrupt.

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]).

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.

Micro-op: Micro-op. Instructions have variable-length encoding and many perform multiple primitive operations. The processor does not execute these complex instructions directly, but, instead, decodes them internally into simpler fixed-length instructions called macro-ops. Processor schedulers subsequently break down macro-ops into sequences of even simpler instructions called micro-ops, each of which specifies a single primitive operation.

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.

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.

Node: A node, is an integrated circuit device that includes one to 8 cores (one or two Core Complexes).

NTA: Non-Temporal Access.

OW: Octword. An 128-bit value.

PCICFG: The PCICFG, or PCI defined configuration space, physical register mnemonic format is of the form DXFYxZZZ.

PCIe®: PCI Express.

PEC: Packet error code.

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.

Processor: A package containing one or more Nodes. See Node.

PTE: Page table entry.

QW: Quadword. A 64-bit value.

REFCLK: Reference clock. Refers to the clock frequency (100 MHz) or the clock period (10 ns) depending on the context used.

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-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.
SMAF: System Management Action Field. This is the code passed from the SMC to the processors in STPCLK assertion messages.
SMI: System management interrupt.
SMM: System Management Mode.
SMT: Simultaneous multithreading. See Core::X86::Cpuid::CoreId[ThreadsPerCore].
SP3: Server, four die MCM, single and dual socket.
SP3r2: High performance desktop, 4-die, single socket, lidded LGA.
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.
Token: A scheduler entry used in various DF queues to track outstanding requests.
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.
VDD: Main power supply to the processor core logic.
VID: Voltage level identifier.
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.
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.
XBAR: Cross bar; command packet switch.

Memory Map - MSR

Physical Mnemonic	Namespace
0000_0000h...0000_0001h	MCA::LS
0000_0010h...0000_02FFh	Core::X86::Msr
0000_0400h...0000_0403h	MCA::LS
0000_0404h...0000_0407h	MCA::IF
0000_0408h...0000_040Bh	MCA::L2
0000_040Ch...0000_040Fh	MCA::DE
0000_0414h...0000_0417h	MCA::EX
0000_0418h...0000_041Bh	MCA::FP
0000_041Ch...0000_043Bh	MCA::L3
0000_043Ch...0000_0443h	MCA::UMC
0000_044Ch...0000_044Fh	MCA::PB
0000_0450h...0000_0457h	MCA::CS
0000_0458h...0000_045Bh	MCA::PIE
C000_0080h...C000_0410h	Core::X86::Msr
C000_2000h...C000_2009h	MCA::LS
C000_2010h...C000_2016h	MCA::IF
C000_2020h...C000_2029h	MCA::L2
C000_2030h...C000_2036h	MCA::DE
C000_2050h...C000_2056h	MCA::EX
C000_2060h...C000_2066h	MCA::FP
C000_2070h...C000_20E9h	MCA::L3
C000_20F0h...C000_210Ah	MCA::UMC
C000_2130h...C000_2136h	MCA::PB
C000_2140h...C000_2159h	MCA::CS
C000_2160h...C000_2169h	MCA::PIE
C001_0000h...C001_029Bh	Core::X86::Msr
C0010400	MCA::LS
C0010401	MCA::IF
C0010402	MCA::L2
C0010403	MCA::DE
C0010405	MCA::EX
C0010406	MCA::FP
C001_0407h...C001_040Eh	MCA::L3
C001_040Fh...C001_0410h	MCA::UMC
C0010413	MCA::PB
C001_0414h...C001_0415h	MCA::CS
C0010416	MCA::PIE
C001_1002h...C001_103Ch	Core::X86::Msr

Memory Map - SMN

Physical Mnemonic	Namespace
0005_9800h: SMUTHMx00000000	SMU::THM