MCA Enhancements in Future Intel® Xeon® Processors

June 2013

Reference Number: 329176-001, Revision: 1.0

Contents

Figures

Tables

Revision History

Document Number	Revision Number	Description	Date
329176-001	1.01	• Initial Release	April 2013

§

CHAPTER 1 MCA ENHANCEMENTS IN FUTURE INTEL® XEON® PROCESSORS

1.1 INTRODUCTION

This white paper describes the software architecture and software flows associated with Enhanced MCA Logging and IO MCA.

1.1.1 References

Intel® 64 and IA-32 Architectures Software Developer's Manual Volumes 3A, 3B and 3C: System Programming Guide

ACPI 5.0 Specification, www.acpi.info

UEFI Specification 2.3.1, www.uefi.org

1.1.2 Definition of Terms

The following includes definition of terms and acronyms used in this document.

Table 1-1.Definitions

Term	Definition
CMCI	Corrected Machine Check Interrupt. Details can be found in Section 15.5 of Volume 3B of the Intel® 64 and IA-32 Architectures Software Developer's Manual.
DIMM	Dual In-line Memory Module.
Enhanced MCA Logging	Refers to improved Firmware First signaling and enhanced error logs that complement machine check bank content. This feature is supported in future generations of Intel® Xeon® Processors.
Intel® QPI	Intel® QuickPath Interconnect.
IO MCA	This feature allows IO errors to be reported via MCA mechanism.
MCA	Machine Check Architecture. Chapter 15 of Volume 3B of the Intel® 64 and IA-32 Architectures Software Developer's Manual describes the Machine Check Architecture supported by processors supporting Intel Architecture.
MCE	Machine Check Exception. Machine Check Exception and associated error code architecture are described in Chapter 15 and 16 of Volume 3B of the Intel® 64 and IA-32 Architectures Software Developer's Manual.
Uncore	Refers to the functionality in processor socket other than processor cores. Uncore encompasses Intel QPI support logic, memory controller and so forth.

1.2 ENHANCED MCA LOGGING

Enhanced MCA Logging allows firmware to provide additional error information to system software, synchronous with MCE or CMCI. MCE and CMCI are described in Chapter 15 of Volume 3B of the Intel® 64 and IA-32 Architectures Software Developer's Manual.

1.2.1 Usage Model

Today's system software relies on machine check bank registers to determine the source of the error. Machine check banks are implemented in hardware and can log a very limited amount of information about the error.

Certain usages such as Predictive Failure Analysis (PFA) require more information about the error than what can be described in processor machine check banks. Most server processors log additional information about the error in processor uncore registers. Since the addresses and layout of these registers vary widely from one processor to another, system software cannot readily make use of them. To complicate matters further, some of the additional error information cannot be constructed without detailed knowledge about platform topology. Enhanced MCA Logging allows firmware to provide additional error information to MCE/CMCI handler and thus addresses this gap.

The following section illustrates the benefit of Enhanced MCA Logging in the context of memory PFA. It should be noted that the system software can benefit from Enhanced MCA Logs in many other ways. For example, the additional information can be stored in system software Event logs or it can be used to suggest a corrective action.

1.2.1.1 Enhanced MCA Logging and Memory PFA

Certain system software products monitor the health of the memory subsystem and are able to take corrective actions to reduce the likelihood of an uncorrectable error. This ability is known as memory PFA. Machine check banks generally capture the Physical Address of the failing location. However, memory PFA algorithm needs to know about physical location (DIMM address) of the failure in order to precisely record and take corrective action. The DIMM address is typically specified in terms of a DIMM serial number and rank/bank/row/column number inside the DIMM. Constructing a DIMM address from Physical address is a relatively complex process that needs to take into account many variables such as memory DIMM internal layout, SMBUS routing on the motherboard and memory address interleaving scheme.

This translation process cannot be readily implemented in processor hardware because of the following reasons.

1. Complexity
2. Platform to Platform variation

Enhanced MCA Logging allows this algorithm to be implemented in platform specific firmware. Platform firmware code can intercept the memory errors, compute the DIMM address and supply it to the system software via Enhanced MCA Logging Interface.

In addition to the DIMM address, PFA requires more details about the cause of the error. For example, PFA software may want to take different actions for DIMM correctable error and memory channel correctable error as shown in the table below. PFA software may be unable to differentiate between the two purely based on machine check register contents since both are logged as ordinary correctable errors in the machine check bank. However, with the help of Enhanced MCA Logging, platform firmware can provide more details about the cause of the error and thus enable richer PFA functionality.

Table 1-2.Likely Actions by Memory PFA Agent

Correctable Memory Error Cause	Likely Memory PFA Agent Action
Memory DIMM correctable errors	Evaluate if the memory page containing the failing location needs to be off-lined.
Transient Memory Channel Error	Ignore.

1.2.2 Discovering support for Enhanced MCA Log

In order to discover support of Enhanced MCA Log, software is required to verify if the processor capability is available. If the processor reports that the capability exists, then it should check if the underlying BIOS supports reporting Enhanced MCA logs for software use. The platform supports Enhanced MCA Log if MCG_ELOG_P capability is set in the processor (see Section 1.2.2.1) and BIOS/firmware declares supports via an Intel specific ACPI DSM method (see Section 1.2.2.2).

Table 1-3.Enhanced MCA Log Discovery

MCG_ELOG_P (IA32_MCG_CAP[26])	Does platform support Enhanced MCA Log?	Enhanced MCA Log DSM	System Software Behavior
0	NA. Platform cannot support Enhanced MCA Log.	Don't care.	Concludes that Enhanced MCA Log is not supported. Shall not invoke Enhanced MCA Log DSM.
1	No.	BIOS does not report Enhanced MCA L1 Directory Pointer. This can be accomplished either by not implementing DSM or by indicating Index 1 is not implemented.	Invokes Enhanced MCA Log DSM if present, but concludes that Enhanced MCA Log Support is lacking.
1	Yes.	Index 1 returns Enhanced MCA L1 Directory Pointer.	Invokes Enhanced MCA Log DSM in order to discover Enhanced MCA L1 Directory Pointer.

1.2.2.1 Discovering Processor Capability

Bit 26 of IA32 MCG_CAP MSR indicates whether the processor supports Enhanced MCA Logging. When MCG_ELOG_P is set, it indicates that the processor allows platform firmware to be invoked when an error is detected so that it may provide additional platform specific information that augments the data included in machine check banks. See Table 1-4.

Table 1-4.Enhanced MCA Logging Capability

64 27	26	25	24 0
Reserved	MCG_ELOG_P	Reserved	See Figure 15-2 of Software Developer's manual Volume 3B.

1.2.2.2 Discovering Platform Capability

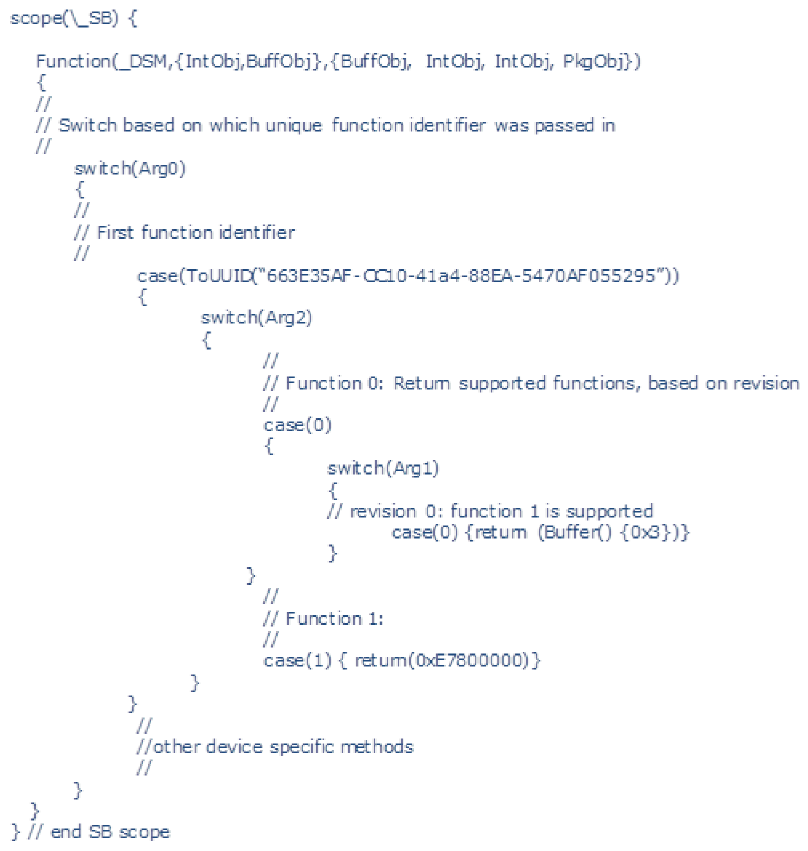
Enhanced MCA Logging is an optional platform capability. Enhanced MCA Logging capable firmware provides additional information about the error to the system software by logging them in memory. System software can traverse the data structures in the memory and locate the Enhanced Error Log. The memory ranges used for these data structures is pre-allocated and reserved by firmware during boot time. As a result, System software can construct linear address mapping at boot time and use that mapping later.

BIOS reports support for Enhanced MCA Logging via an Intel specific ACPI DSM under the scope _SB. See ACPI Specification for definition of DSM and _SB. The Enhanced MCA Logging Device Specific Method is identified with a GUID shown in Table 1-5. If implemented, the Index function 1 returns the pointer to Enhanced MCA Level 1 (L1) Directory. The format of Enhanced MCA L1 Directory is described in Section 1.2.3.

Table 1-5.Enhanced MCA Logging DSM

GUID	Revision	Index	Description
663E35AF-CC10-41a4-88EA-5470AF055295	0	0	Query: If the platform supports Enhanced MCA Logging, the query function returns 3 and indicate index 1 is implemented. If the platform does not support Enhanced MCA Logging, the query function returns 0 to indicate index 1 is not implemented.
	0	1	Implemented if the platform supports Enhanced MCA Logging. Evaluates to an integer and returns 64 bit address of Enhanced MCA L1 Directory. The address must be 4K aligned and points into Firmware reserved memory. The address can be greater than 4 GB. System software should cross-check this ad-dress against number of Physical address bits supported by the processor and ensure it points into Firmware Reserved Memory and is 4K aligned. If these checks fail, System software should not make use of Enhanced MCA Logging. The Enhanced MCA L1 directory pointer cannot change during a given boot and the firmware must guarantee that the return value will stay the same for a given boot. System software is free to invoke this function only once every boot and cache the response.

Sample Code



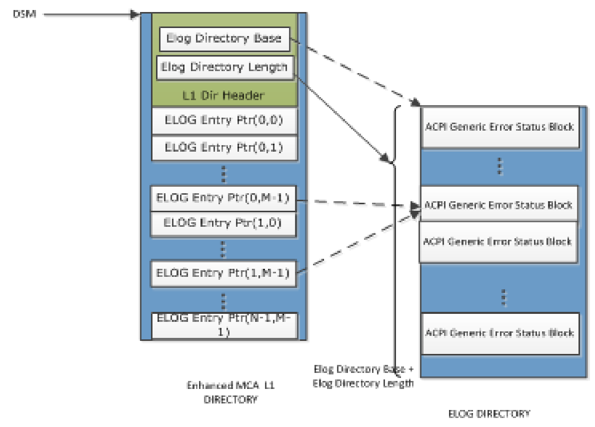
```
scope(\_SB) {

   Function(_DSM,{IntObj,BuffObj},{BuffObj,  IntObj, IntObj, PkgObj})
   {
   //
   // Switch based on which unique function identifier was passed in
   //
       switch(Arg0)
       {
       //
       // First function identifier
       //
              case(ToUUID("663E35AF-CC10-41a4-88EA-5470AF055295"))
              {
                     switch(Arg2)
                     {
                            //
                            // Function 0: Return supported functions, based on revision
                            //
                            case(0)
                            {
                                   switch(Arg1)
                                   {
                                   // revision 0: function 1 is supported
                                          case(0) {return (Buffer() {0x3})}
                                   }
                          }
                            //
                            // Function 1:
                            //
                            case(1) { return(0xE7800000)}
                     }
             }
              //
              //other device specific methods
              //
       }
   }
} // end SB scope
```

1.2.3 Memory Data Structures

Index 1 of Enhanced MCA Logging DSM returns the pointer to Enhanced MCA L1 Directory. Enhanced MCA L1 Directory contains a header followed by a number of pointers to Error Log data structures contained in Elog Directory. Various Enhanced Memory Log data structures and their relationship are shown in Figure 1-1. M represents the number of L1 Directory Entries per logical processor. N represents the highest possible value of APIC ID. The following sections describe format of Enhanced MCA L1 Directory and Error Log Directory in details.

Figure 1-1. Enhanced MCA Error Log Data Structures


DSM
Elog Directory Base
Elog Directory Length
L1 Dir Header
ELOG Entry Ptr(0,0)
ELOG Entry Ptr(0,1)
ELOG Entry Ptr(0,M-1)
ELOG Entry Ptr(1,0)
ELOG Entry Ptr(1,M-1)
ELOG Entry Ptr(N-1,M-1)
Enhanced MCA L1 DIRECTORY
Elog Directory Base + Elog Directory Length
ACPI Generic Error Status Block
ACPI Generic Error Status Block
ACPI Generic Error Status Block
ACPI Generic Error Status Block
ELOG DIRECTORY

1.2.3.1 Enhanced MCA Level 1 Directory

Enhanced MCA L1 Directory structure starts with a header and is followed by list of Error Log (ELOG) Directory Entry Pointers. All addresses in these data structures represent host physical addresses.

Machine check banks can be shared between logical processors. The sharing scheme varies with processor model. Each entry in Elog Directory corresponds to one Physical Machine Check bank. Enhanced MCA L1 Directory provides a level of indirection between the logical processor's view of machine check banks and the physical machine check banks.

This is illustrated in Figure 1-1. Bank M-1 is shared between Processor 0 and Processor 1. Hence these two entries in L1 Directory point to one and the same Error Log Directory entry.

The format of L1 Directory is shown below. All numbers are in decimal format.

Table 1-6.Enhanced MCA L1 Directory Format

Field	Byte Length	Byte Offset	Description
Header			
Version	4	0	Header Version in major.minor format. This field is set to 0100 to represent version 1.00.
Header Length	4	4	Length, in bytes, of this header. For version 1.00, this field shall contain the value 64.
L1 Directory Length	8	8	Length, in bytes, of the entire L1 Pointer Directory including this header.
Elog Directory Base	8	16	This is the base address of Error Log Structure Directory.
Elog Directory Length	8	24	Length, in bytes, of the entire Error Log Directory.

Table 1-6.Enhanced MCA L1 Directory Format

Field	Byte Length	Byte Offset	Description
Flags	4	32	Bit 0 - System software Opt-in. Enhanced MCA Log capable BIOS initializes this flag to 0. An Enhanced MCA Log aware System software opts into Enhanced MCA Logging by setting this flag to 1 during initialization. BIOS may be capable of signaling certain errors via two paths - Enhanced Machine check architecture or via ACPI Generic Hardware Source. If this flag is set, BIOS will surface such errors via Enhanced MCA Architecture mechanism. If this flag is clear, BIOS may surface such errors via alternate mechanism. Bits 31:1 are reserved.
Reserved	12	36	Reserved
Number of L1 Entries per Logical Processors	4	48	The number of L1 directory entries per logical processor.
Reserved	12	52	Reserved
Elog Directory Entry Pointers[*n*]	-	64	This field is 8 bytes in size. 63 62 0 | Valid | ElogPointer | If Valid (Bit 63) is set, bits 62:0 contain the physical address of corresponding valid ACPI Generic Error Status Block structure. Each such structure must be 4KB aligned. If Bit 63 is not set, bits 62:0 are invalid. These pointers are pre-populated by the BIOS at boot time for all logical processors that are present.

System software can use the base and the length fields in Enhanced MCA L1 Directory Header to allocate memory if it wants to maintain a local copy of Error Log structures.

The memory range that contains Enhanced MCA L1 directory is pre-allocated by BIOS at boot time and is declared as part of Firmware Reserved memory. The size of L1 directory is large enough so as to accommodate processors that may be hot added.

Enhanced MCA L1 directory header is followed by an array of Elog Directory Entry Pointers. System software can locate the Elog Directory Entry by indexing into this Array. The index value corresponding to machine bank i on processor with X2APIC ID x is governed by the equation.

Index = x * Number of L1 Entries per Logical Processors + i

Number of L1 Entries per Logical Processors is derived from offset 48 of the L1 Directory Header. The first machine check bank, i=0 always starts at MSR address 400H. Machine check bank i starts at MSR address 400H+4 * i.

1.2.3.2 Error Log Directory

Error Log (Elog) Directory is a contiguous data structure. It contains a number of entries, each of which is in the form of an ACPI Generic Error Data structure. Each Elog Directory Entry will contain exactly one Generic Error Status Block Structure. Table 1-7 describes interpretation of various Generic Error Status Block fields when used in Enhanced MCA context. The memory range that contains Elog Directory is pre-allocated by BIOS at boot time and is declared as part of Firmware Reserved memory. The size of Elog directory is large enough so as to accommodate processors that may be hot added.

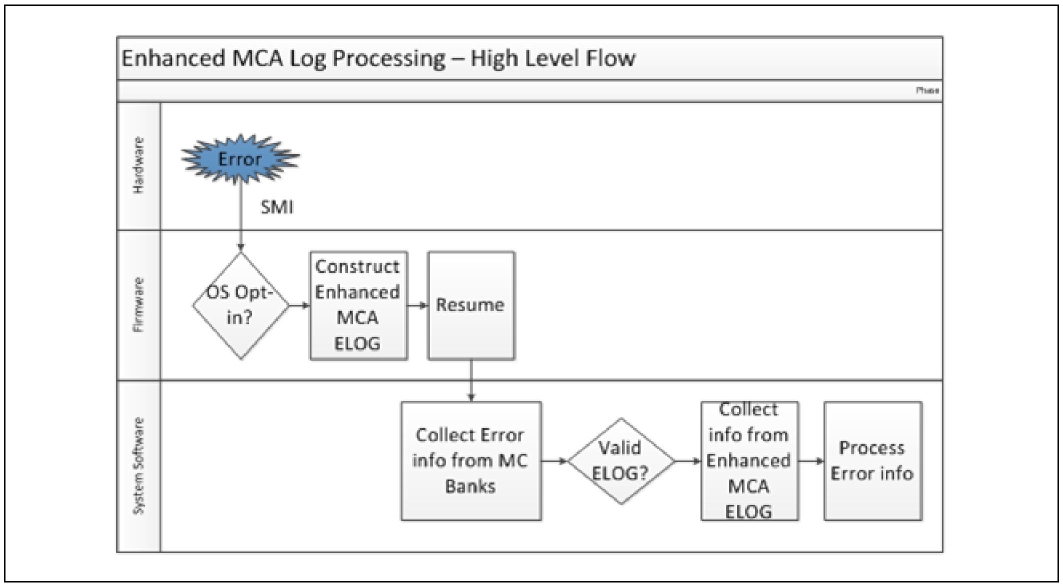
Table 1-7.Interpretation of ACPI Generic Error Status Block Fields

Field	Byte Length	Byte Offset	Description
Block Status	4	0	Indicates the type of error information reported in the error packet. The bit definition is consistent with ACPI 5.0 specification. This field is also used for record management. The usage is consistent with ACPI 5.0 specification. If the field contains a 0, it indicates that record is either invalid or is consumed by the system software. System software should not attempt to access this record. Platform Firmware can update the contents of the record when Block Status is 0. If the field contains a value other than 0, it indicates that Platform Firmware has placed a valid data in MCA Elog, but system software has not consumed it. Platform Firmware can choose to overwrite the Elog, but must follow MCA overwrite rules if it does.
Raw Data Offset	4	4	Offset in bytes from the beginning of the Error Status Block to raw error data, per ACPI 5.0 specification.
Raw Data Length	4	8	Length in bytes of the raw data, per ACPI 5.0 specification.
Data Length	4	12	Length in bytes of the generic error data, per ACPI 5.0 specification.
Error Severity	4	16	Identifies the error severity of the reported error. Enhanced MCA usage is consistent with ACPI Specification.
Generic Error Data Entries	Data Length	20	In the context of Enhanced MCA Error log, the information contained in this field is a collection of one or more Generic Error Data Entries. The format of Generic Error Data Entry is defined in ACPI 5.0 Specification.

1.2.4 Software Flows

The combined firmware and software flow is shown in Figure 1-2. As shown, the hardware generates an SMI upon error. The SMI handler pre-processes the error and constructs Error Log in memory prior to continuing with the MCE or CMCI.

Figure 1-2. Combined Flow


Enhanced MCA Log Processing – High Level Flow
Phase
Hardware
Firmware
System Software
Error
SMI
OS Opt-in?
Construct Enhanced MCA ELOG
Resume
Collect Error info from MC Banks
Valid ELOG?
Collect info from Enhanced MCA ELOG
Process Error info

The subsequent sections describe the system software Initialization flow and system software Runtime flow.

1.2.4.1 Enhanced MCA Log Initialization

Enhanced MCA Log aware system software will execute the following steps during MCA initialization.

1. Determine whether the processor supports Enhanced MCA Log feature by reading MCG_ELOG_P capability bit in IA32_MCG_CAP MSR. If MCG_ELOG_P is 0, skip Enhanced MCA Log Initialization and do not make use of Enhanced MCA Log feature.
2. If Enhanced MCA Log DSM method (Section 1.2.2.2) is not present, skip Enhanced MCA Log Initialization and do not make use of Enhanced MCA Log feature.
3. If Enhanced MCA Log DSM method (Section 1.2.2.2) is present, invoke index 0. If bit 1 is not set in the return value, skip Enhanced MCA Log Initialization and do not make use of Enhanced MCA Log.
4. Invoke index 1. If the return value is greater than the physical address space of the processor or does not point to an address inside Firmware Reserved Address Space or is not 4K aligned, skip Enhanced MCA Log Initialization and do not make use of Enhanced MCA Log.
5. If Index 1 returns a valid pointer, de-reference it to locate L1 Directory.
6. Parse L1 Directory Header. System software can statically allocate sufficient memory to hold the necessary data structures and compute linear addresses corresponding to these physical addresses.
7. Set system software Opt-in flag in L1 Header.

1.2.4.2 Runtime Flow

Enhanced MCA Logging aware system software will execute the following steps during MCE/CMCI handling if it detected Enhanced MCA Log support during initialization:

1. Use existing algorithms to select one or more logical processors that can query machine check banks for error status.
2. The logical processor that is processing the error determines its own X2APIC ID (or XAPIC ID if it is in legacy XAPIC mode). Let's say the APIC ID is A.
3. Use existing algorithms to locate the MCA bank(s) associated with the MCA or CMCI event. Let's

say this is bank number B.

4. Initialize ELOG_FOUND=FALSE.
5. If Enhanced MC Logging is not supported by platform, fall back to legacy MCA handling.
6. Locate the L1 Dir Entry corresponding to this <A,B> pair. This L1 Dir Entry is at the address [L1 Dir Base + L1 Dir Header Size + 8 * (A * Number of L1 Entries per Logical Processors + B)]. This process may involve physical address to linear address translations, unless system software is using pre-translated addresses.
7. If Elog Dir Entry Pointer is not valid (bit 63=0), assume that there is no enhanced Log for this error event. System software should proceed with legacy MCA handling.
8. If Elog Dir Entry Pointer is valid (bit 63=1), Bits 62:0 of L1 Dir Entry has the ad-dress of the Elog Entry. Validate this pointer to ensure it points into firmware reserved address range.
9. Read first DWORD of the Elog entry.
   a. If the field contains a value other than 0, it indicates that BIOS has placed a valid data in MCA Elog.
      i. Copy the record to its local buffers and clear the first DWORD of Elog Entry.
      ii. Set ELOG_FOUND=TRUE to ensure Enhanced MCA Log is processed along with MCA Bank contents. The Enhanced MCA Log contents may not be consistent with MCA Bank in case of an overwrite condition.
   b. If the field contains a 0, there is no MCA error log record. Fall back to legacy MCA handling.

1.3 IO MCA

Logging and Signaling of errors from PCI Express domain is governed by PCI Express Advanced Error Reporting (AER) architecture. PCI Express architecture divides errors in two categories: Uncorrectable errors and Correctable errors. Uncorrectable errors can further be classified as Fatal or Non-Fatal. Uncorrected IO errors are signaled to the system software either as AER Message Signaled Interrupt (MSI) or via platform specific mechanisms such as NMI. Generally, the signaling mechanism is controlled by BIOS and/or platform firmware. IO MCA is a processor error handling mode where Uncorrected PCI Express errors are signaled in the form of machine check exception and logged in machine check banks.

When processor is in IO MCA mode, Uncorrected PCI Express errors are logged in the MCACOD field of the IA32_MCi_STATUS register as Bus and Interconnect compound error code of 0000_1110_0000_1011 (0x0E0B). Refer to volume 3, section 15.9.2.5 in Intel® 64 and IA-32 Architectures Software Developer's Manual for encoding of Bus and Interconnect Errors. Machine check logging complements and does not replace AER logging that occurs inside the PCI Express hierarchy. The PCI Express Root Complex and Endpoints continue to log the error in accordance with PCI Express AER mechanism. In IO MCA mode, MCi_MISC register in the bank that logged IO MCA can optionally contain information that link the Machine Check logs with the AER logs. In such scenario, the machine check handler can utilize the contents of MCi_MISC to locate the PCI Express AER logs corresponding to the same error. Specifically, if MCi_Status.MISCV is 1 and MCACOD is 0x0E0B, MCi_MISC contains the PCI Express address of the Root Complex device containing the AER Logs. Software can consult the header type and class code registers in the Root Complex device's PCIe Configuration space to determine what type of device it is. Errors that originate from PCI Express or Legacy Endpoints are logged in the corresponding Root Port in addition to the generating device. In this case, MCi_MISC contains the address of the Root Port if MISCV=1 and software can parse the Root Port AER logs to learn more about the error.

The format of MCi_MISC for IO MCA errors is shown in Table 1-8.

Table 1-8.Meaning of MCi_MISC when MCACOD is 0x0E0B and MISCV=1

63:40	39:32	31:16	15:9	8:6	5:0
Reserved	PCI Express Segment Number[1]	PCI Express Requestor ID[2]	Reserved	ADDR MODE[3]	RECOV ADDR LSB[4]

Notes:

1. Refer to PCI Firmware Specification 3.0 for an explanation of PCI Express Segment Number and how software can access configuration space of a PCI Express Device given the segment number and Requestor ID.
2. Refer to PCI Express Specification 3.0 for definition of PCI Express Requestor ID and AER architecture.
3. Not Applicable if ADDRV=0
4. Not Applicable if ADDRV=0