Understanding VMware ESXi PSOD Machine Check Exceptions

Planning / Implementation

VMware ESXi PSOD stands for Purple Screen of Diagnostics (also known as Purple Screen of Death). The name is derived from Microsoft Windows stop error screen, known as the Blue Screen of Death. The diagnostics screen appears when VMware ESXi kernel detects a fatal error.

Intel and AMD implement Machine-Check Architecture (MCA) that detects and reports hardware issues, including system bus errors, RAM (ECC and parity) errors, and other CPU-related faults.

MCE is a critical error that occurs when a computer's processor detects when it identifies a serious hardware fault. These exceptions are typically caused by memory corruption, CPU cache failures, or other hardware faults. There are a set of model-specific registers (MSRs) that are used to report errors. When a hardware error occurs, global and bank-specific status machine-check architecture registers are populated with information regarding the cause, and whether the CPU can safely continue execution. In the case of a correctable error, ESXi logs the incident and register contents in the VMkernel logs. If an error is uncorrectable, and the CPU cannot continue safely, ESXi halts with a purple diagnostic screen.

An example PSOD message is shown in the following figure.

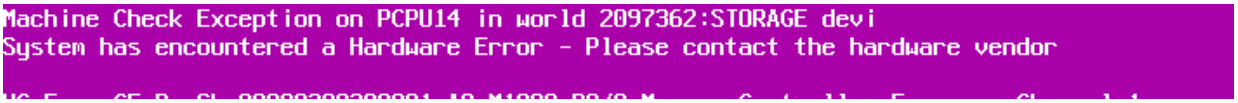

Figure 1. Example PSOD

To diagnose the issue, capture a screenshot of the PSOD screen. Then, reboot the system, and collect the logs.

Machine-Check Architecture Registers

Machine-Check Architecture (MCA) Registers are part of Intel's system used to log machine-check errors. These specialized hardware registers that collect error information when a MCE occurs. Key registers include MCI_STATUS, MCi_ADDR, and MCi_MISC registers, which provide insights into the type and location of hardware errors. The global MCA register (MCG_STATUS) reports whether an MCE is in progress.

The log message consists of one line for each relevant bank and includes the physical detail such as

- The physical CPU number
- The text "MCA:" label
- The error class
- How the error was reported
- The MCG_STATUS register (G)
- The bank number (B)
- The MCi_STATUS register (S)
- The MCi_ADDR register (A)
- The MCi_MISC register (M)
- The decoded system physical address and size (P) (applicable in 6.7 and later)
- A human-readable interpretation of the error.

Example Logs:

```
cpu…)MCA: … : UC Excp G5 B3 Sbe00000000080189 A8e70096740 Mb07485 Cache Hier
archy: Level 1 Cache Snoop Error.
cpu…)MCA: … : SRAR Excp Gf B1 Sbd80000000100134 Aa83bf95200 M86 Pa83bf95200/
40 Cache Hierarchy: Level 0 Data Cache DataRead Error.
```

Error Classes:

- UC: Uncorrected, unrecoverable
- SRAR: Uncorrected, recoverable, action required (Intel)
- SRAO: Uncorrected, recoverable, action optional (Intel)
- UCNA: Uncorrected, no action required (Intel)
- UCR: Uncorrected, recoverable (AMD)
- CE: Corrected
- DE: Deferred (AMD)

Error Reporting Methods:

- Init: Found during boot-time initialization (possibly from prior to the reboot)
- Poll: Periodic polling of the MCA banks
- Excp: MCE handler
- Intr: Corrected Machine Check Interrupt handler

Automatic Interpretation

VMware ESXi attempts to interpret status register contents for display in the log and the purple diagnostic screen. For example:

- Cache Hierarchy: Level 0 Data Cache Read Error.
- Bus error, node originated, read, memory access

In this section:

Decoding the global MCA status (MCG_STATUS) register

The MCG_STATUS register plays a crucial role in Intel's Machine Check Architecture (MCA) by providing a high-level summary of the processor's state after a machine-check exception (MCE) occurs. This 64-bit register helps determine whether execution can safely resume and whether the recorded error is directly tied to the instruction pointer.

This section explains the significance of these bits and provides an example analysis of how to interpret the register's value during an error event.

The following figure illustrates the structure of the MCG_STATUS register, but only the lower 4 bits are defined, each serving a specific diagnostic purpose. These bits indicate whether execution can restart, if an error is linked to the instruction pointer, and whether an MCE has been recorded.

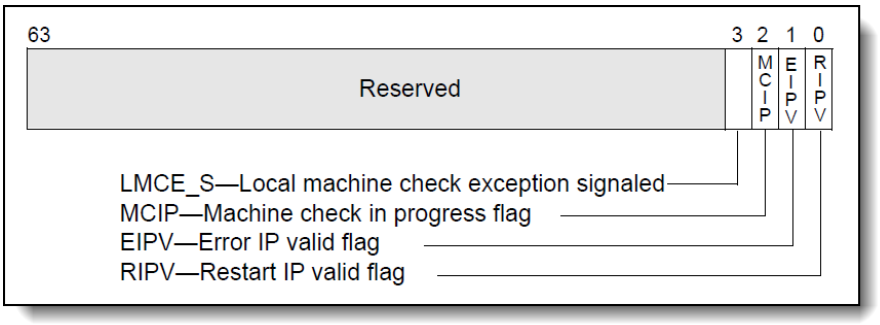

Figure 2. MCG_STATUS Register (from Intel)

The global status register is 64-bit, but only the low 4 bits have meaning:

- **Bit 0 (RIPV)** — Indicates that execution can restart reliably
- **Bit 1 (EIPV)** — Indicates that the instruction pointer is associated with the error.
- **Bit 2 (MCIP)** — Indicates an MCE was generated
- **Bit 3 (LMCE_S)** — Indicates a local MCE.

Example Analysis:

```
cpu…)MCA: … : UC Excp G5 B3 Sbe00000000080189 A8e70096740 Mb07485 Cache Hier
archy: Level 1 Cache Snoop Error.
```

If the global status register value is "5" (0101 in binary), this indicates to:

- LMCE_S=0
- MCIP=1
- EIPV=0
- RIPV=1

This indicates a machine check is in progress, and the Restart IP is valid.

Manual Decoding of MCi_STATUS register

The MCi_STATUS register is a 64-bit model-specific register (MSR) that provides detailed error reporting when a machine-check exception (MCE) occurs. This register helps diagnose hardware failures by indicating the validity, severity, and source of detected errors

This section explains the significance of each field within the MCi_STATUS register and how they contribute to system debugging and error handling.

The following figure illustrates the structure of the MCi_STATUS register, where the high-order bits (57:63) summarize the processor state and provide key diagnostic information, such as whether the error is uncorrectable, if multiple errors have occurred, and whether the processor state may be corrupted. Additionally, the lower bits contain error codes that can be used to further analyze the nature of the fault.

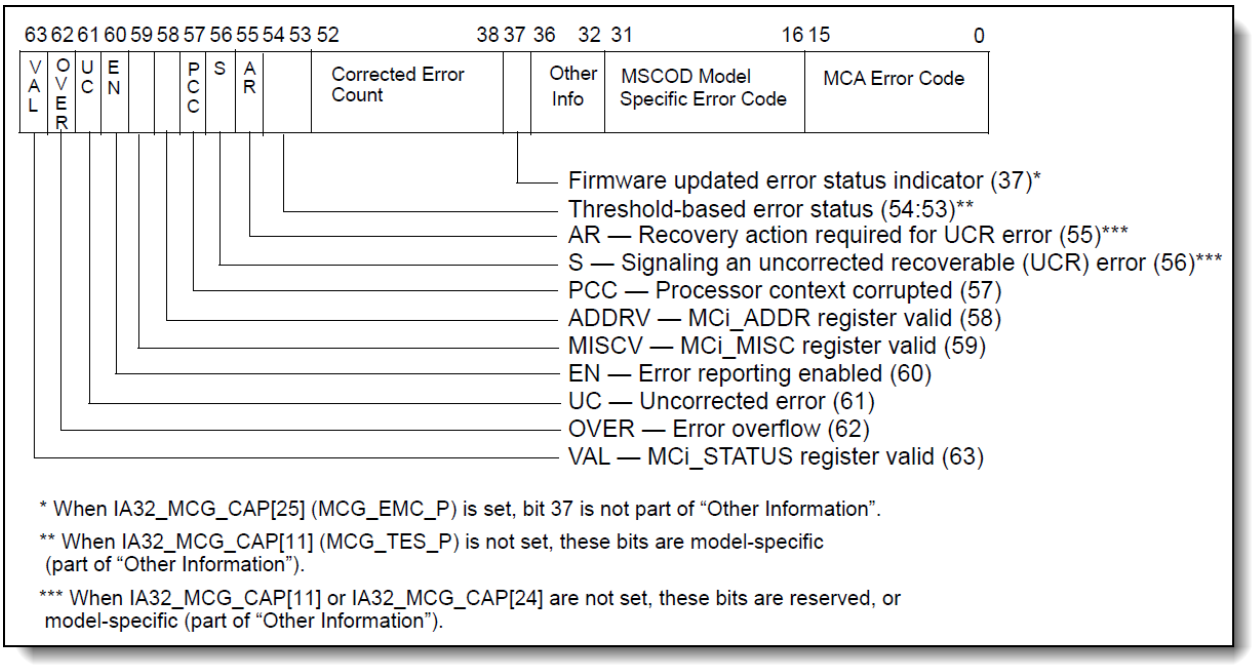

Figure 3. MC*i*_STATUS Register (from Intel)

The MCi_STATUS register (64 bits) provides additional error details. The high-order bits (57:63) summarize the processor state:

- **Bit 63 (VAL):** Indicates that the register contains valid data.
- **Bit 62 (OVER):** Indicates multiple errors occurred before reporting.
- **Bit 61 (UC):** Uncorrectable error.
- **Bit 60 (EN):** Error reporting was enabled.
- **Bit 59 (MISCV):** Miscellaneous register contains valid data.
- **Bit 58 (ADDRV):** Address register contains valid data.

- **Bit 57 (PCC):** Processor state may be corrupted.
- **Bits 56:32**: contain other information, which may be reserved, used for counters, or hold other information that is model-specific. For more information, see the vendor documentation listed in the reference of this article
- **Bits 31:16** contain a model-specific extended error code. For more information, see the vendor documentation listed in the reference of this article
- **Bits 15:0**: contains the machine-check architecture-defined error code for the machine-check error condition detected.

Machine-check architecture-defined error codes

The lower 16 bits (0:15) define several errors, categorized into Simple and Compound error codes.

Option 1: Using Automatic Tool

You can debug the error efficiently using the following tool which automates the steps below:

https://vmware-psod-reader.github.io/vmware-psod-reader/

Option 2: Using Manual Steps

The manual steps refer to the process of manually decoding machine-check architecture (MCA) error codes by interpreting their binary encoding. This method is useful when automated tools are unavailable, allowing engineers and system administrators to directly diagnose hardware failures from the MCi_STATUS register values.

MCA error codes help identify various types of processor and memory-related errors. These errors can be categorized into **simple errors** and **compound errors**, as shown in the two tables below.

Table 1. Simple Error Codes

Error Code	Binary Encoding	Meaning
No Error	0000 0000 0000 0000	No error has been reported to this bank of error-reporting registers.
Unclassified	0000 0000 0000 0001	This error has not been classified into the MCA error classes.
Microcode ROM Parity Error	0000 0000 0000 0010	Parity error in internal microcode ROM
External Error	0000 0000 0000 0011	The BINIT# from another processor caused this processor to enter machine check.
FRC Error	0000 0000 0000 0100	FRC (functional redundancy check) main/secondary error.
Internal Parity Error	0000 0000 0000 0101	Internal parity error.
SMM Handler Code Access Violation	0000 0000 0000 0110	An attempt was made by the SMM Handler to execute outside the ranges specified by SMRR.
Internal Timer Error	0000 0100 0000 0000	Internal timer error.
I/O Error	0000 1110 0000 1011	Generic I/O error.
Internal Unclassified	0000 01xx xxxx xxxx	Internal unclassified errors.

Table 2. Compound Error Codes

Type	Form	Interpretation
Generic Cache Hierarchy	000F 0000 0000 11LL	Generic cache hierarchy error
TLB Errors	000F 0000 0001 TTLL	{TT}TLB{LL}_ERR
Memory Controller Errors	000F 0000 1MMM CCCC	{MMM}_CHANNEL{CCCC}_ERR
Cache Hierarchy Errors	000F 0001 RRRR TTLL	{TT}CACHE{LL}_{RRRR}_ERR
Extended Memory Errors	000F 0010 1MMM CCCC	{MMM}_CHANNEL{CCCC}_ERR
Bus and Interconnect Errors	000F 1PPT RRRR IILL	BUS{LL}_{PP}_{RRRR}_{II}_{T}_ERR

The list below describes the encoding of the placeholder fields (such as {MMM}, {CCCC}, {TT}, {LL}, {RRRR}, {PP}, and {II}) in the Table 2.

- Encoding of Transaction Type (TT) sub-field:
  - 00 – Instruction
  - 01 – Data
  - 10 – Generic
  - 11 – Reserved
- Encoding of Memory Hierarchy Level (LL) sub-field:
  - 00 – Level 0
  - 01 – Level 1
  - 10 – Level 2
  - 11 – Generic
- Encoding of memory transaction type (MMM) sub-field:
  - 000 – Generic undefined request
  - 001 – Memory read error
  - 010 – Memory write error.
  - 011 – Address or command error.
  - 100 – Memory scrubbing error.
  - 101-111 – Reserved.
- Encoding of channel number (CCCC) sub-field:
  - 0000-1110 – Channel number.
  - 1111 – Channel not specified.
- Encoding of Request (RRRR) sub-field:
  - 0000 – Generic error
  - 0001 – Generic read
  - 0010 – Generic write
  - 0011 – Data read
  - 0100 – Data write
  - 0101 – Instruction fetch
  - 0110 – Prefetch
  - 0111 – Evict
  - 1000 – Snoop (probe)
- Encoding of Participation Processor (PP) sub-field:
  - 00 – Local node originated the request.
  - 01 – Local node responded to the request.
  - 10 – Local node observed error as third-party.
  - 11 – Generic
- Encoding of Timeout (T) sub-field:
  - 0 – Request did not timeout.
  - 1 – Request did timeout.

- Encoding of Memory/IO (II) sub-field:
  - 00 – Memory access
  - 01 – Reserved
  - 10 – I/O
  - 11 – Other

Example Analysis of an MCE PSOD

The figure below is an example PSOD screen, illustrating how an error is reported when a failure is detected in the memory controller. By converting the MCi_STATUS register value to binary and analyzing its lower bits, we can determine the specific error type and affected hardware component.

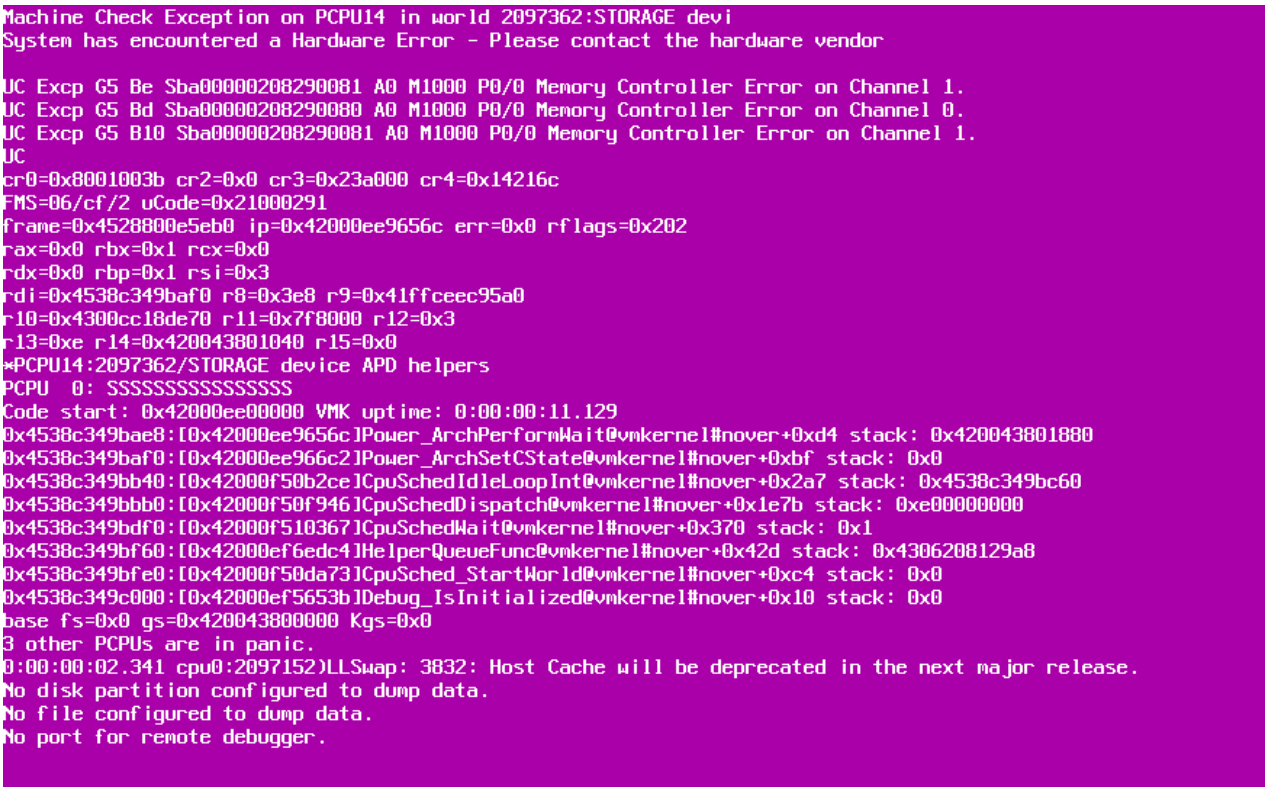

Figure 4. Example PSOD screen

Given an MCi_STATUS register value of **0xba00000288290081**. Convert it to binary:

```
1011 1010 0000 0000 0000 0000 0000 0010 0000 1000 0010 1001 0000 0000 1000 0
001
```

Extracting the lower 16 bits (`0000 0000 1000 0001`), we identify:

- Memory controller errors
- Generic undefined request
- Channel 1 error

VMware's PSOD screen would show: "Memory Controller Error on Channel 1”

After replacing the processor with an embedded memory controller, the issue was resolved

Reference

For more information, see these resources:

- Intel - Chapters 17 and 18 of the Intel 64 and IA-32 Architectures Software Developer's Manual: https://cdrdv2.intel.com/v1/dl/getContent/671200
- AMD - Chapter 9 of the AMD64 Architecture Programmer's Manual, Volume 2: System Programming: https://www.amd.com/content/dam/amd/en/documents/processor-tech-docs/programmer-references/24593.pdf
- VMware KB, Decoding Machine Check Error (MCE) output after an ESXi panic (Purple Screen): https://knowledge.broadcom.com/external/article/367928/decoding-machine-check-error-mce-output.html
- Nutanix KB, Debugging ESX Machine Check Exception (MCE) PSOD: https://portal.nutanix.com/page/documents/kbs/details?targetId=kA0600000008gYECAY

Author

David Hsia is an OS Engineer in the Lenovo Infrastructure Solutions Group, based in Taipei, Taiwan. As a specialist in Linux and VMware technical support, he is interested in operating system and focuses on VMware vSphere and ESXi.

Thanks to the following specialists for their contributions and suggestions:

- Chengcheng Peng, Lenovo VMware Engineer
- Alpus Chen, Lenovo VMware Engineer
- Skyler Xing12 Zhang, Lenovo VMware Engineer
- David Watts, Lenovo Press Senior Manager

Related product families

Product families related to this document are the following:

- VMware vSphere

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service. Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
8001 Development Drive
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION ”AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary. Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk. Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This document, LP2176, was created or updated on March 5, 2025.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
  https://lenovopress.lenovo.com/LP2176
- Send your comments in an e-mail to:
  comments@lenovopress.com

This document is available online at https://lenovopress.lenovo.com/LP2176.

Trademarks

Lenovo and the Lenovo logo are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. A current list of Lenovo trademarks is available on the Web at https://www.lenovo.com/us/en/legal/copytrade/.

The following terms are trademarks of Lenovo in the United States, other countries, or both:
Lenovo®

The following terms are trademarks of other companies:

AMD is a trademark of Advanced Micro Devices, Inc.

Intel® is a trademark of Intel Corporation or its subsidiaries.

Microsoft® and Windows® are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.