



Learn the Architecture - RAS Software Guide

Version 1.0

Non-Confidential

Copyright © 2024 Arm Limited (or its affiliates).
All rights reserved.

Issue 03

109579_100_03_en



Learn the Architecture - RAS Software Guide

Copyright © 2024 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
0100-03	15 October 2024	Non-Confidential	First release.

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm

makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

- 1. Introduction to RAS..... 6**
 - 1.1 Error signaling..... 6
 - 1.2 RAS reporting flow..... 6
 - 1.3 Error handling software..... 7
- 2. Firmware First error handling.....8**
 - 2.1 CPER..... 8
 - 2.2 APEI..... 8
 - 2.3 SDEI.....9
- 3. Kernel First error handling.....10**
 - 3.1 AEST..... 10
 - 3.1.1 Processor component structure..... 10
 - 3.1.2 Memory controller component structure..... 10
 - 3.1.3 SMMU component structure..... 10
 - 3.1.4 Vendor defined component structure..... 11
 - 3.1.5 GIC component structure..... 11
 - 3.1.6 Interfaces..... 11
 - 3.1.7 Interrupts..... 11
- 4. Virtual machines..... 12**
 - 4.1 Firmware First..... 12
 - 4.2 Kernel First..... 12
- 5. Related information..... 13**

1. Introduction to RAS

Reliability, Availability, and Serviceability (RAS) aims to increase the robustness of a system by detecting hardware errors, recording them and correcting them where possible. Arm's RAS extension provides this robustness to both the processor and system architectures.

RAS defines components, such as Nodes and Error Records, and terminology around handling and containing errors. The RAS Overview explains the terms used in this guide. [RAS Overview](#)

The RAS architecture defines hardware features to detect and signal that errors have occurred. Software support is needed to handle those errors once they have been signaled.

This guide describes the software flows for recording and handling RAS errors.

1.1 Error signaling

When a RAS node detects an error, based on the type and correctability of the error software needs to be notified. The error can be signaled in the following ways:

- In-band error response, also known as an External abort
- Fault Handling Interrupt (FHI)
- Error Recovery Interrupt (ERI)
- Critical Error Interrupt (CRI)

Typically in-band errors, FHI and ERI are handled by the Application Processor (AP) in the system, either by the Operating System or a combination of the OS and Firmware.

CRIs could indicate that the AP is not in a safe state to continue so these are typically not handled by the AP but by a System Control Processor (SCP).

The rest of this document will only discuss in-band responses, FHI and ERI.

1.2 RAS reporting flow

The propagation of a detected RAS error is as follows:

1. Errors detected by a component are reported to a node.
2. The node stores details about the detected error in one or more error records. The error record includes information such as the severity of the error, affected addresses and a count of previously corrected errors.
3. If possible the component corrects the error, which will allow system progress to continue. If the error is uncorrectable then this can be reported immediately or the error can be deferred until the corrupted data is consumed by creating poison. If the error is detected on a read then

the return value is tagged as being poisoned. On a write the poison is propagated on to the target.

4. Corrected and deferred errors do not require software intervention. Software may wish to be notified so that the cause of the error can be investigated. This is done by generating an FHI.
5. Uncorrected errors require software intervention. If not deferred they can be reported to software by returning an in-band error response or raising an ERI. An in-band response is more common if the error was detected as the direct side-effect of a memory access. It is possible for a node to generate both types of notification.
6. FHI and ERI are received on the CPU as normal interrupts and are routed to the appropriate piece of handling software. In-band errors will be signaled through the memory system reporting channel and will generate either an SEA or SEI depending on the CPU implementation and configuration.

1.3 Error handling software

RAS support is necessary in both the firmware and the Rich Operating System due to the way RAS errors are usually signaled.

A software module called Operating System-directed configuration and Power Management (OSPM) manages RAS errors.

There are two ways to deliver RAS errors to the OSPM:

Firmware First handling delivers the error notification to the firmware to be handled initially. This allows the fault interrupt to be configured as Secure. Firmware then creates a data structure to contain the error record and delivers that to the Rich Operating System running in Non-secure state. The [UEFI Specification](#) defines a data structure called Common Platform Error Record (CPER). The CPER is then shared with the OSPM.

Kernel First handling delivers the error notification directly to the OS. This means that the RAS interrupt must be delivered as a Non-secure interrupt. In systems where a Non-secure hypervisor is present this will first receive error notifications on behalf of the OS.

2. Firmware First error handling

Firmware First error handling initially reports the RAS error to firmware running at EL3.

Firmware is responsible for reading the RAS error record provided by the hardware and transposing the information into a CPER data structure.

In the case of error reporting, such as FHI or ERI, firmware may choose not to notify the OSPM. Firmware may just create a record in memory and use a doorbell interrupt to notify higher level software.

For SEA or SEI where an error has been consumed, firmware will notify the OS running at a lower exception level. The Software Delegated Exception Interface (SDEI) can then notify the OSPM that the error occurred and the location of the CPER. [SDEI Specification](#)

2.1 CPER

A CPER is a standard data structure used to communicate error information to the OSPM.

The CPER consists of:

- A header that uniquely identifies a RAS error record
- A number of section descriptors that contain error description data

2.2 APEI

APEI is a standard method for delivering error information from firmware to the OSPM. It uses a series of standard tables to describe errors, one of which is the Hardware Error Source Table (HEST).

The HEST consists of:

- A header that identifies the creator and OEM that is defining these error sources
- Some error source descriptors

Error source descriptors can be used to describe a variety of hardware error reports, including some architecture specific errors and different types of PCIe errors. RAS errors in Arm systems are described as Generic Hardware Error Source version 2 (also known as Type 10) descriptors.

The Generic Hardware Error Source version 2 descriptor includes information that describes the error source to the system, including:

- A unique ID
- If the source is enabled

- The address of a register containing the address of the memory where the error status data can be found
- Additional information about the address space used to store the error data
- Notification type (Interrupt, NMIO, SEI etc.)
- Error threshold information - the number of interrupts within a time period that constitute an error

2.3 SDEI

SDEI is a software mechanism for delivering system events from higher exception levels to lower exception levels. Use cases for SDEI include:

- Firmware interrupts
- System watchdog timers
- RAS errors

SDEI is used to deliver RAS errors from firmware to the Non-secure OS or Hypervisor. As exceptions routed to EL3 will pre-empt lower exception levels the Interface allows events to be considered at a higher priority than OS or Hypervisor interrupts due to them being initially delivered to EL3.

Client software, such as the OS or Hypervisor, registers a conduit with the firmware SDEI dispatcher. Registration is done by making an SMC.

SDEI events are delivered from EL3 by firmware performing an exception return to a predefined context defined by the conduit. As this is initiated from a higher EL it cannot be masked by lower Exception Levels.

3. Kernel First error handling

In this configuration external aborts generated by RAS errors that are available to the Non-secure software are routed directly to the kernel.

3.1 AEST

The Arm Error Source Table (AEST) describes the hardware error sources present in the system to the OSPM. This information allows the OSPM to register an interrupt handler for each error node that it supports. [ACPI for Armv8 RAS](#) describes the AEST format.

The AEST consists of a header providing vendor and OEM identification and a number of AEST nodes. Each AEST node describes an error source. Each AEST node entry describes a specific component type and contains information about that component's interfaces and any interrupts it generates.

3.1.1 Processor component structure

This type is used to describe error nodes related to a processor. The data structure provides the ACPI Processor ID that the component belongs to. Its affinity level and information about the resource type.

Resource types are:

- Cache
- TLB
- Generic

For cache or TLB entries the level of cache, and if the resource is shared or not, are also encoded.

3.1.2 Memory controller component structure

The structures contain the System Resource Affinity Table (SRAT) Proximity domain of the resource.

3.1.3 SMMU component structure

System Memory Management Units (SMMUs) can also have multiple error nodes associated with them such as TLBs and internal configuration caches. The AEST node contains a reference to an IO Remapping Table (IORT) entry for the SMMU component.

3.1.4 Vendor defined component structure

This node type allows unique hardware types to be registers and linked to a vendor specific driver using generic data encoded in this structure.

3.1.5 GIC component structure

ACPI terminology refers to a Generic Interrupt Controller (GIC) as an Advanced Programmable Interrupt Controller and each controller is identified by an entry in a Multiple APIC Description Table (MADT). This structure encodes an index into the MADT that identifies the GIC instance that is associated with this error node.

3.1.6 Interfaces

The interfaces section of the AEST node describes if the node uses a System Register (SR) or MMIO interface to present its error record. The structure also records how many error records are present and if the node supports polling.

3.1.7 Interrupts

The entry reports information about the interrupt type and the interrupt ID and if the node can generate either of these RAS interrupt types:

- ERI
- FHI

CRI are not supported by ACPI so cannot be reported through the method.

4. Virtual machines

If the operating system is running inside a Virtual Machine (VM) then the mechanism for notifying the OS that a RAS error has been reported differs. The hypervisor handles the error and then determines how the Guest OS is notified.

4.1 Firmware First

The Virtual Machine Manager (VMM) is responsible for configuring the system to use a Firmware First handling scheme. In this case the VMM is responsible for notifying the Guest OS in a similar way to how the firmware notifies a non-virtualized OS.

The hypervisor notifies the VMM that a RAS error has occurred. Under Linux/KVM the notification is via signals. The VMM converts the information to CPER and sends an ACPI notification.

4.2 Kernel First

Kernel first handling of RAS errors usually relies on the error being reported by the hardware being exposed directly to the OS. If a VM is in use then SEA and SEI exceptions will usually be routed to EL2 to be handled by the Hypervisor.

The VMM creates the AEST and manages how the Guest OS interacts with the virtualized hardware. The hypervisor can emulate MMIO hardware RAS error nodes to allow the Guest OS the collect error information.

5. Related information

Here are some resources related to material in this guide:

- [Learn the Architecture RAS Overview](#) provides an overview of the RAS Architecture.
- [Arm Architecture Reference Manual for A-profile architecture](#) provides the specification for the RAS processor architecture.
- [ACPI specification landing page](#) contains links to the ACPI specifications and other resources.
- [Software Delegated Exception Interface](#) contain more information on SDEI.