AMPERE®

Ampere® Altra® Max 64-Bit Multi-Core Processor User's Manual

July 17, 2023

Document Issue 1.15

July 17, 2023

Ampere Computing reserves the right to change or discontinue this product without notice.

This document is the Ampere Computing® Altra® Max User's Manual. Make sure you are using the correct edition for the level of the product.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

The information contained in this document is subject to change or withdrawal at any time without notice and is being provided on an "AS IS" basis without warranty or indemnity of any kind, whether express or implied, including without limitation, the implied warranties of non-infringement, merchantability, or fitness for a particular purpose.

Any products, services, or programs discussed in this document are sold or licensed under Ampere Computing's standard terms and conditions, copies of which may be obtained from your local Ampere Computing representative. Nothing in this document shall operate as an expressed or implied license or indemnity under the intellectual property rights of Ampere Computing or third parties.

Without limiting the generality of the foregoing, any performance data contained in this document was determined in a specific or controlled environment and not submitted to any formal Ampere Computing test. Therefore, the results obtained in other operating environments may vary significantly. Under no circumstances will Ampere Computing be liable for any damages whatsoever arising out of or resulting from any use of the document or the information contained herein.

Ampere Computing
4655 Great America Parkway, Santa Clara, CA 95054
Phone: (669) 770-3700
https://www.amperecomputing.com

Ampere Computing reserves the right to make changes to its products, its datasheets, or related documentation, without notice and warrants its products solely pursuant to its terms and conditions of sale, only to substantially comply with the latest available datasheet.

Ampere, Ampere Computing, the Ampere Computing and 'A' logos, and Altra are registered trademarks of Ampere Computing.

Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All other trademarks are the property of their respective holders.

Contents

CHAPTER 8 Reliability, Availability, and Serviceability (RAS)

CHAPTER 9 Clocks and Reset

CHAPTER 10 Boot Process

Figures

Tables

About This Book

This introduction describes the purpose and audience for this manual, along with conventions, additional documentation, contact information for Ampere Computing®, and the revision history of the manual.

Purpose

The Ampere® Altra® Max Architecture provides the foundation for a new generation of flexible processing nodes required to satisfy the demands of server systems, intelligent networks, and pervasive embedded computing applications. The Altra Max Architecture provides a platform for a wide range of applications including cloud computing, data center servers, enterprise servers, communications systems, and other mission-critical systems.

Additional details, along with functional timing and mechanical information, can be found in device-specific datasheets (see *"Additional Documentation"*).

Audience

This document is intended for system designers and software developers who plan to build a system using Altra Max devices, and for people who want to understand Altra Max processor functions and capabilities.

Conventions

All numbers are decimal unless they have one of these prefixes:

- 0x prefixes a hexadecimal number.
- 0b prefixes a binary number.

These names are used:

- Halfword – Two bytes.
- Word – Four bytes.
- Doubleword – Eight bytes.
- Quadword – 16 bytes.

Additional Documentation

Additional documentation is available from Ampere Computing at https://connect.amperecomputing.com/, including:

- Altra Max Processor Product Briefs.
- Altra Max Processor Datasheets.

How to Contact Ampere Computing

Ampere Computing
4655 Great America Parkway, Santa Clara, CA 95054
Phone: (669) 770-3700
https://www.amperecomputing.com
Email: support@amperecomputing.com

Revision History

This table outlines the revision history of this manual.

Revision History (Sheet 1 of 3)

ISSUE	DATE	DESCRIPTION
1.15	July 17, 2023	Added: • *Section 11.6.3, "Current Limits."* Minor technical and editorial revisions.
1.10	June 2, 2023	Added: • *Section 13.6.3, "Cluster Processor Module (CPM) Debug Map."* • *Section 13.6.4, "Core 0 and Core 1 Debug Map."* • *Section 13.6.5, "Debug Trace Controller (DTC) Debug Map."* Updated: • *Section 2.1, "Overview."* • *Figure 2-2: "CPM Block Diagram."* • *Section 11.3.1, "System-Level Idle Management."* • *Section 11.3.1.3, "System Standby State."* • *Section 12.3, "Shared Peripheral Interrupts (SPIs)."* • *Figure 13-1: "Debug Subsystem Block Diagram."* • *Table 13-2: "External and Self-Hosted Debug Address Map."* • *Section 13.6, "Debug Address Maps,"* to divide existing content into *Section 13.6.1, "Altra Max (Armv8) DAP Address Map," Section 13.6.2, "Debug Address Map,"* and *Section 13.6.6, "Internal Debug APB-AP Address Map."* Minor technical and editorial revisions.

Revision History (Sheet 2 of 3)

ISSUE	DATE	DESCRIPTION
1.05	July 25, 2022	Updated these sections to revise descriptions of PCIe and CCIX: • *Section 1.2, "Altra Max Processor Summary."* • *Section 1.6, "PCI Express (PCIe)."* • *Section 5.2, "Features."* • *Section 5.3, "PCI Express (PCIe) Root Complexes (RCs)."* • *Section 5.6, "Cache Coherent Interconnect for Accelerators (CCIX) Controllers."* Minor technical and editorial revisions.
1.00	May 1, 2022	• Changed "master" to "primary" and "slave" to "secondary" throughout. • Added: – *Section 2.2.3, "Cache Topology."* • Updated: – *Section 1.6, "PCI Express (PCIe)."* – *Section 2.4, "System Level Cache (SLC)."* – *Section 2.3.2.1, "Non-Uniform Memory Access (NUMA)."* – *Table 6-2: "2P System Address Space."* – *Table 8-3: "RAS Hardware Error Handling."* – *Figure 8-1: "Firmware Boot-Time Hardware Error Handling."* – *Section 8.3.2, "Firmware Boot-Time Hardware Error Handling."* – *Table 8-4: "HEST Error Sources."* – *Section 13.6, "Debug Address Maps."* Minor technical and editorial revisions.
0.65	October 1, 2021	• Updated: – *Table 11-7: "CPPC Performance Thresholds."* – *Table 11-8: "Altra Max CPPC Registers."* – *Section 11.4.1, "Collaborative Processor Performance Control (CPPC)."* • Deleted *Section 11.5, "Maximum Frequency Mode."* • Removed the term "maximum frequency mode" throughout; in some cases the term is replaced by "maximum/nominal frequency." • Minor technical and editorial revisions.

Revision History (Sheet 3 of 3)

ISSUE	DATE	DESCRIPTION
0.60	June 18, 2021	• Changed product name from Mystique to Altra Max. • Updated *Section 5.2, "Features,"* to reflect the absence of PCIe P2P support. • Added *Section 5.7, "Ampere Link Interconnect (ALI)."* • Added *Section 8.5.6, "Ampere Link Interconnect (ALI) Error Record Registers."* • Extensively revised *Chapter 13, "Debug and Trace"*: – Updated *Section 13.1, "Overview."* – Added *Section 13.2, "Debug Hardware Components."* Some items in this section were previously included in *Section 13.1, "Overview."* – Replaced Section 13.3, "Joint Test Action Group (JTAG) Debug Interfaces," with *Section 13.3, "Debug Subsystem Diagrams."* – Replaced *Section 13.4, "Armv8 Debug Domains,"* with *Section 13.4, "Joint Test Action Group (JTAG) Debug Interfaces."* – Replaced *Section 13.5, "Processor Life Cycle State (LCS),"* with *Section 13.5, "Debug Security."* – Replaced *Section 13.6, "Debug Control Unit (DCU) eFuses,"* with *Section 13.6, "Debug Address Maps."* • Minor technical and editorial revisions.
0.50	June 11, 2020	Initial issue.

Overview

1

This chapter provides an overview of the Altra® Max 64-Bit Multi-Core Processor. The chapter covers these topics:

1.1 Terminology

Some of the terms in this section appear in labels for blocks in *Figure 1-1*, the Altra Max processor block diagram.

Note: An Altra Max processor is a System-on-Chip (SoC). However, to avoid ambiguity throughout this manual, the terms *SoC*, *SoC logic*, and *SoC agents* describe logic outside the Processor Complex (PCP).

- *Altra Max processor*: The entire chip, which implements the Altra Max Architecture. It comprises the PCP and SoC power domains shown in *Figure 1-1*.
- *Processor*: Same as Altra Max processor.
- *Altra Max core*: One CPU in an Altra Max processor, with Level 1 (L1) Instruction Cache (i-cache) and Data Cache (d-cache), and Level 2 (L2) cache.
- *Core*: An Altra Max core, unless otherwise specified. Also called a CPU or Processing Element (PE).
- *CPU*: Same as core.
- *CPM*: Cluster Processor Module. Each CPM contains two Altra Max cores.
- *PCP*: The PCP contains CPMs, the Coherent Mesh Interconnect (CMI), System Level Cache (SLC), and Memory Controller Units (MCUs).
- *SoC*: The part of the Altra Max processor that contains Input/Output (IO) agents, memory-mapped registers, and microcontrollers. It connects to the PCP through the CMI.
- *SMpro*: System Management Processor, a microcontroller in the SoC domain of an Altra Max processor. It is used for boot control and other firmware functions.
- *PMpro*: Power Management Processor, a microcontroller in the SoC domain of an Altra Max processor. It is used for power management and other firmware functions.
- *Coherence Domain*: Cache coherence in the PCP domain.
- *Agent*: A PCP unit connected to the CMI, or an SoC unit connected to a Peripheral Component Interconnect Express (PCIe) Root Complex (RC) or to an interface in the system block (labeled SYS in *Figure 1-1*). Agents operate as primary or secondary devices, or, for some agents, as both.
  - *PCP Agents*: The CPMs, MCUs, and the SLC.
  - *SoC Agents*: An IO controller or microcontroller that connects to an RC or the system block.

In *Figure 1-1*, the MCUs appear to be separate from the PCP because the figure organizes the Altra Max processor by power domain, and the MCUs are in the SoC power domain despite being part of the PCP.

For definitions of abbreviations and acronyms used in this book, see *Appendix A, "Glossary."*

1.2 Altra Max Processor Summary

Designed to meet the requirements of modern datacenters, Ampere Altra® Max delivers predictable performance, high scalability, and power efficiency for data center deployments from hyperscale cloud to the edge cloud. Altra Max processors provide server-class performance, power management, Reliability, Availability, and Serviceability (RAS), and security. Efficient power management enables innovative system designs to support high-performance, efficient applications. Integrated PCIe controllers support the 64-bit cores and enable Single-Socket Platform (1P) and Dual-Socket Platform (2P) board designs. The Altra Max Architecture provides server-class features, such as end-to-end data protection and performance monitoring.

An Altra Max processor includes:

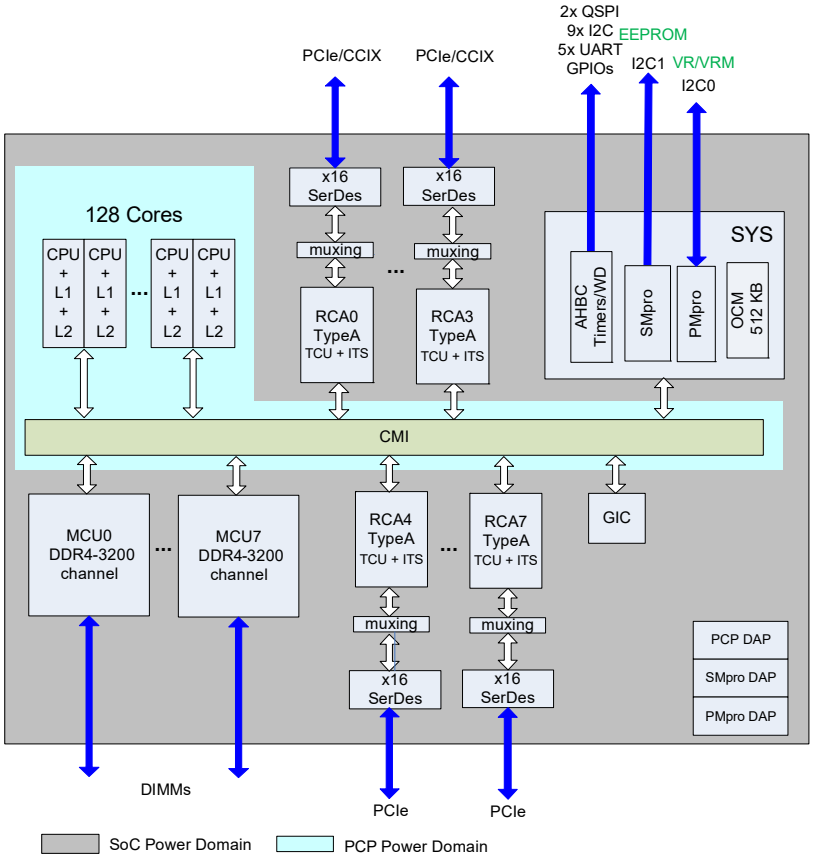
- Up to 128 single-thread Armv8 cores, arranged in pairs in 64 CPMs. Each core has a 64 KB Level 1 (L1) Instruction Cache (i-cache) and 64 KB L1 Data Cache (d-cache), along with a 1 MB Level 2 (L2) cache.
- Cores implement the 64-bit (Aarch64) execution state. The 32-bit (AArch32) execution state is also implemented, but EL1 AArch32 support is removed while EL0 AArch32 support is retained.
- A PCP, with up to 64 dual-core CPMs, and a CMI containing a 16 MB SLC distributed across CMI nodes.
- Up to eight MCUs, for up to eight DDR4-3200 ECC memory channels.
- Up to 16 DIMMs (two DIMMs per memory channel, or 2DPC) for up to 4 TB of memory per processor socket.
- Symbol-based Error-Correcting Code (ECC), which detects and corrects a x4 DRAM chip failure, along with standard Single Error Correction and Double Error Detection (SECDED).
- Arm® Server Base System Architecture (SBSA) Level 4 Firmware compliance.
- SMpro and PMpro support for secure boot and programmable Advanced Configuration and Power Interface (ACPI)-compliant power management features.
- Cache-coherent protocol for PCP communications.
- 128 lanes of PCIe Gen4 connectivity (192 lanes in 2P configurations):
  - Up to 32 controllers to support x16, x8, and x4 links.
  - Native hot-plug support for x4 links.
  - Four dual-mode PCIe/Cache Coherent Interconnect for Accelerators (CCIX) x16 links. In a 2P system, two CCIX x16 links per processor connect the two Altra Max processor sockets.
- Low-speed interfaces include Inter-Integrated Circuit (I^2C), Quad Serial Peripheral Interface (QSPI), Universal Asynchronous Receiver/Transmitter (UART), General-Purpose Input/Output (GPIO), and General-Purpose Input (GPI).
- System counter: One Arm-compliant system counter.
- Generic timers: The two generic timers comply with the Arm specification. There are four timer frames and one control base. The control base contains two secure frames.
- Watchdog timers: There are two watchdog timers, one in the normal world and the other is in the secure world.
- Arm Generic Interrupt Controller (GIC) v3-compliant interrupt controller.
- IO virtualization using SMMU v3.
- Debug and trace support.

Altra Max processors are designed to run applications such as web front-ends and web server middleware, Memcached and in-memory databases, RAID 5 cold storage, and big-data applications using Hadoop and MapReduce. The processors support these applications using specialized power management agents.

Data movement on an Altra Max processor is performed in read/write transactions among Altra Max cores, SoC agents, and memory. A doorbell mechanism manages communication among on-chip intelligent components, such as the Altra Max cores and the SMpro and PMpro microcontrollers. An Altra Max processor also implements fair sharing of peripheral interconnect, with credit management between agents.

Figure 1-1 provides a detailed block diagram of an Altra Max processor. For details, see *Chapter 2, "Processor Complex (PCP)."*

Figure 1-1: Altra Max Processor Block Diagram



1.3 Cluster Processor Module (CPM)

A CPM comprises two Altra Max cores. An Altra Max processor can contain up to 64 CPMs (128 cores). The cores conform to the 64-bit Armv8 architecture. CPMs connect to the rest of the Altra Max processor through the CMI.

The CPMs support the Arm privilege hierarchy of exception levels, from EL0 to EL3 For the Aarch64 execution state, where EL3 is the most privileged. (In the Aarch32 execution state, the Altra Max cores support only EL0.)

The Altra Max cores support the ARM64, ARM32, and T32 (Thumb) instruction sets. The 64-bit and legacy 32-bit ISAs are fully compatible. The ARM ISA supports these operations: integer; scalar; floating-point including half-, single-, double-precision, and Fused Multiply-Add (FMADD); and 128-bit Single Instruction, Multiple Data (SIMD).

The Altra Max cores provide server-class reliability features, such as end-to-end data poisoning, error isolation, and static and dynamic power management features. The Altra Max cores support full virtualization of Altra Max cores, system, interrupts, and timers, along with standard performance monitoring and extensive debug features.

For details, see *Section 2.2, "Cluster Processor Module (CPM)."*

1.4 Processor Complex (PCP)

The PCP contains these communicating agents:

- Up to 64 CPMs.
- The CMI.
- Eight DDR4-3200 SDRAM MCUs and eight memory channels; each channel provides two slots.
- One shared SLC (up to 16 MB).

The eight memory channels are 64 bits wide (72 bits with ECC) and support up to 16 DIMMs, for up to 4 TB of system memory per socket.

The PCP supports the Altra Max Communication Protocol, a cache-coherent protocol for coherent and non-coherent communications among the agents. This protocol is implemented by and on top of the CMI in the PCP.

For details, see *Chapter 2, "Processor Complex (PCP)."*

1.5 Virtualization

Virtualization consolidates multiple guest OS servers onto one Altra Max processor. This load-balances hardware utilization and reduces power consumption and management overhead. Virtualization supports Virtual Machine (VM) portability and management.

An Altra Max Processor provides hardware acceleration for up to 256 active VMs. Each guest OS may or may not be aware of the underlying hypervisor layer. A guest OS running in a VM can access only a specific region of system memory, which the hypervisor defines as Virtual Address (VA) spaces. Each guest OS maps its VAs to Altra Max processor PAs. Each guest OS can independently access hardware resources assigned to it by the hypervisor and does not interfere with hardware resources assigned to any other guest OS. If a guest OS attempts to access the resources of another guest OS, the system detects such requests and protects the other guest OSs. This protection does not impact other well-behaved guest OSs.

IO virtualization abstracts upper-layer network protocols from their physical interfaces. IO virtualization avoids, as much as possible, hypervisor trapping requests from an Altra Max core to an SoC agent. This results in high-performance PCIe and Interprocessor Communication (IPC) flows. Trapping, which involves snooping, checking, and address translation, slows the system. When a system is virtualized, each VM has its own memory and device spaces that do not collide with other VMs. If a memory region is shared among several VMs, the hypervisor must trap all requests to that region and make the appropriate permission checks and translations.

Arm System Memory Management Unit (SMMU) v3-compliant hardware provides IO virtualization to most interfaces. The PCIe RCs and internal Direct Memory Access (DMA) support IO virtualization. It is possible to oversubscribe an interface because it is expected that the aggregate actual bandwidth is less than the theoretical bandwidth, or because it is known that all VMs are not continuously and simultaneously active.

1.6 PCI Express (PCIe)

Altra Max processors support PCIe networking cards up to 100 Gigabit Ethernet (GbE) or even faster, along with storage expanders and Non-Volatile Memory Express (NVMe) storage devices, making the PCIe implementation well-suited for big data applications.

An Altra Max processor provides 128 PCIe lanes in a 1P system and 192 PCIe lanes in a 2P system; two x16 PCIe/CCIX RcAs connect each socket to the other. This on-chip connectivity enables communication with other agents, such as the DRAM memory controller, Altra Max cores, or the PCIe ports themselves. Depending upon selected bifurcation options, an Altra Max processor provides up to 32 PCIe controllers.

Altra Max processors support PCI Express Base Specification Revision 4.0, which operates at 2.5 GT/s (gigatransfers per second), 5 GT/s, 8 GT/s, and 16 GT/s. There is support for 32- and 64-bit addressing, with Maximum Payload Size (MPS) of 512 bytes for x16 and x8 controllers and 256 bytes for x4.

The 128 lanes of PCIe connectivity are implemented in eight x16 RCs of Type A (RcA). Four of the RcAs can be implemented as x16 CCIX links. Two of the CCIX links can be used to connect two Altra Max processors in a 2P system.

All PCIe ports are Root Ports (RPs).

For details, see *Chapter 5, "PCI Express (PCIe) Subsystem."*

1.7 Security

Altra Max processors support these security features:

- All features required by SBSA Level 4 – firmware (EL3, secure memory, secure boot).
- Secure world and normal world main memory and Memory-Mapped IO (MMIO) transactions.
- SMpro as the SoC Root of Trust (RoT) for secure transactions.

The Altra Max cores support the Arm privilege hierarchy of exception levels, from EL0 to EL3, where EL3 is the most privileged. EL3 is intended for a monitor that operates at a higher secure level than EL2, where hypervisors and VMs operate; EL1, for secure and guest operating systems; and EL0, for user applications.

The processor implements Arm security technology, which isolates software and hardware states and resources into two "worlds":

- *Secure world*: All storage, state, and functionality that must be protected from accidental or malicious attack lives here. Accesses to memory and devices in this world are permitted onlyby hardware and software that are in the secure world.
- *Normal world*: All other storage, state, and functionality lives here in this world. Hardware and software from both worlds can accesses memory and devices in this world. However, hardware and software in this world cannot access memory and devices in the secure world.

An NS bit associated with all accesses represents the secure (NS=0) and normal (NS=1) worlds. Primary devices in the secure world can generate accesses with NS=0 and NS=1; primary devices in the normal world can generate only accesses with NS=1. Architecturally, the NS bit distinguishes between two 48-bit PA spaces: a secure world address space (NS=0) and a normal world address space (NS=1). In effect, the NS bit acts as an additional address bit but it is not used for address routing. Instead, the NS bit is used mainly to determine permissions for secondary device-side accesses to particular PAs.

For details, see *Chapter 7, "Security"* and *Chapter 10, "Boot Process."*

1.8 SMpro and PMpro Microcontrollers

An Altra Max processor contains these 32-bit Arm Cortex-M3 microcontrollers:

- The *SMpro (System Management)* microcontroller provides overall system management. It provides an interface to the BMC for SoC-BMC communication. When enabled in eFuse Trusted Management Module (TMM) mode, SMpro handles secure boot. SMpro manages SoC clock and reset, system booting, power-failure detection, and error handling. SMpro can communicate with the other processing agents using doorbell interrupts and message-passing.
- The *PMpro (Power Management)* microcontroller provides power management functions, including ACPI Performance State (P-State) implementation. PMpro also offloads extended power management functions such as Dynamic Voltage and Frequency Scaling (DVFS). PMpro monitors die temperatures and sensor interfaces to perform over-temperature protection and dynamic power estimation.

1.9 Low-Speed Interfaces

Altra Max processors provide these low-speed interfaces:

- I^2C: Nine I^2C ports for system software, and two additional I^2C ports for SMpro and PMpro firmware.
- QSPI: Two QSPI primary ports operate at up to 33 MHz for SPI flash and Trusted Platform Module (TPM) connectivity and are configurable as secure world or normal world.
- UART: Five UART ports: one is secure and four are configurable as secure world or normal world.
- GPIO: Three sets of eight GPIOs with interrupt capability. Each set (GPIO0-7, GPIO8-15, and GPIO16-23) can be configured for the secure world or the normal world.
- GPI: Eight GPIs (General Purpose Inputs) are in the normal world and have no interrupt capability. The signal state is readable by software through a Read-Only (RO) register.

1.10 Counter and Timers

An Altra Max processor provides one System Counter (SC), along with these timers:

- Generic Timers: Two Generic Timers comply with the Arm specification. There are four timer frames and one control base. For more information about the timer frames and control base, see *Section 4.3.7, "Timer Frames and Control Base."*
- Watchdog Timers: Two Watchdog Timers, one in the normal world, the other in the secure world.

1.11 Reliability, Availability, and Serviceability (RAS) and Error Handling

Altra Max processors support these RAS features:

- *Reliability*: Parity and ECC protection throughout the chip, including on all caches in the PCP, all DDR main memory, and all RAM macro memories in the SoC. Errors, when first detected, are immediately logged and optionally reported using interrupts. Permanently corrupt data, that is, uncorrectable or unrecoverable by hardware, is marked as poisoned. Corruption cannot propagate to other data computed and stored by a CPU core. Attempted use of poisoned data results in a precise exception. For example, when a load instruction attempts to access corrupt data, a precise exception is taken on that instruction. Only agents that attempt to use corrupt data are affected. For example, if a core takes a software exception in a VM and the OS process is performing a data load instruction, other cores and IO agents remain unaffected.
- *Availability*: End-to-end error propagation of data poisoning, so that consumers of corrupt (poisoned) data receive exceptions wherever they are. Errors are reported as precise in context as possible so that an error handler can determine the affected processing flow. Sufficient error information is logged so that the affected context and scope of the impact can be determined. Correctable ECC is used for large memory structures.
- *Serviceability*: Support for both internal platform management software and remote management. The on-chip programmable SMpro microcontroller can provide remote-management capability independently or in with an external on-board Baseboard Management Controller (BMC). The Armv8 architecture provides a most-privileged level of execution (EL3), above the application, OS, and hypervisor software exception levels at which platform management software typically runs.

Errors are handled separately in CPMs and SoC blocks. In CPMs, hardware detects many errors. Most errors are associated with RAM storage arrays and the corruption of stored bits by soft and/or hard errors. In general, errors are detected, corrected if possible, logged in hardware registers, and optionally reported to software, and an exception is taken in the case of Uncorrectable Errors (UEs). The CPMs and other agents in a PCP conform with the system-wide error-handling architecture.

For details, see *Chapter 8, "Reliability, Availability, and Serviceability (RAS)."*

1.12 Power Management

An Altra Max processor manages its power consumption. The features are managed by the on-chip PMpro power management microcontroller. Each processor block implements various power states. Some states are entered automatically; that is, they are not directly controlled by the user. Users can configure various control registers to select other power states.

PMpro exposes and supports ACPI features, such as Collaborative Processor Performance Control (CPPC) and Low Power Idle (LPI).

Power-management features include:

- ACPI firmware interface.
- Control of an external VRM to support Dynamic Voltage Scaling (DVS) for CPMs.
- Dynamic Frequency Scaling (DFS) of CPMs.
- Adaptive hardware control of voltage and frequency to maximize performance for a given power level.
- Dynamic power estimation.
- DDR self-refresh on a per-rank basis.

- Hardware-initiated power-saving states in the MCUs.
- Numerous temperature sensors and counters for power events.

An Altra Max processor contains two primary power domains that are separate from each other and from a few additional secondary power domains. The primary power domains, in the order in which they are powered up, are:

- SoC power domain (MCUs, GIC, SMpro and PMpro, and SoC peripheral interfaces).
- PCP power domain (CPMs and the CMI).

The Altra Max processor is connected to an external power control device that drives the power supplies and informs the Altra Max processor when the power supplies for both power domains are stable.

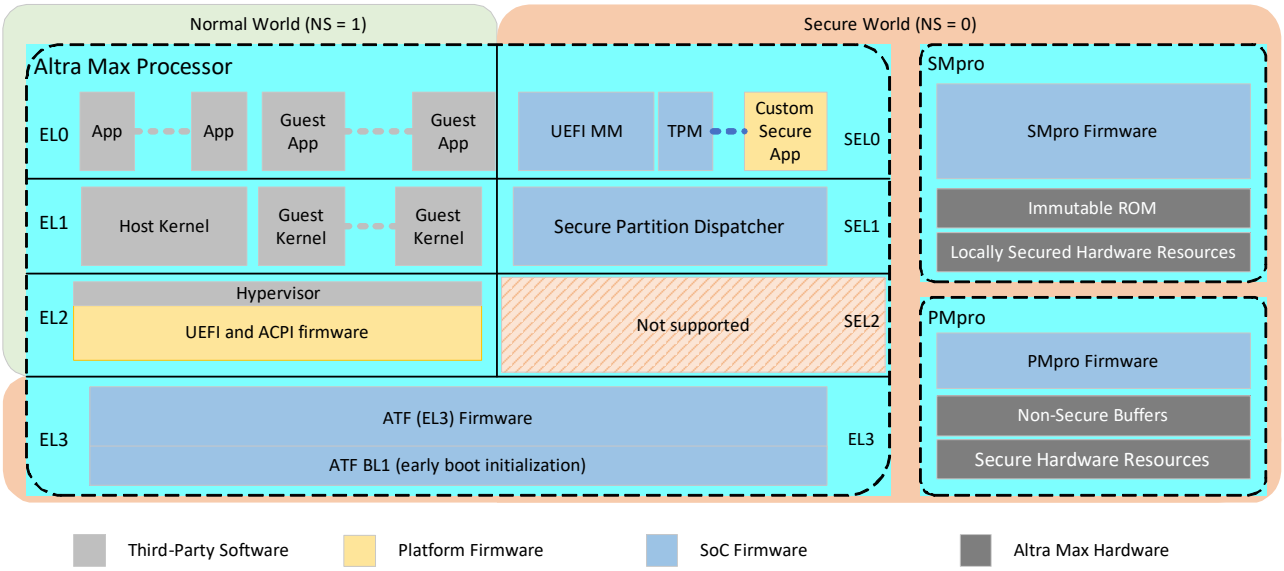
For details, see *Chapter 11, "Power Management."* For detailed power measurements, including Thermal Design Power (TDP), refer to *Altra Max Datasheet*.

1.13 Firmware

Two types of firmware are provided with an Altra Max processor:

- Platform firmware.
- SoC firmware.

Figure 1-2: Altra Max Processor Firmware Architecture



1.13.1 Platform Firmware

Platform firmware refers to:

- Firmware that runs in the normal world (NS=1), including Unified Extensible Firmware Interface (UEFI), ACPI, and NVPARAM images.

Note: The legacy term "Basic Input/Output System (BIOS)" is sometimes used to refer these firmware images.

- Custom secure run-time application images that run in secure exception level 0 (SEL0). This can include run-time platform firmware functionality that is external to UEFI and ACPI, for example, PCIe hot plug firmware.
- Firmware that is owned by the platform owner (an Original Design Manufacturer (ODM), Original Equipment Manufacturer (OEM), or end customer) and can be signed by the platform owner.
  - For the Ampere reference board design: This firmware is owned and signed by Ampere.
  - For all other platforms, this firmware is owned and optionally signed by the platform owner.

Ampere Computing delivers reference platform firmware source code based on the Ampere Altra Max reference platform. The reference platform firmware source code comprises build scripts, tools, and utilities packaged in a "development kit" to enable building custom Altra Max platforms. Platform firmware also includes the SoC firmware binary images.

1.13.2 SoC Firmware

SoC firmware, which is owned by and must be signed by Ampere Computing, refers to:

- Firmware that runs on SMpro and PMpro microcontrollers, including SMpro and PMpro firmware images.

Note: The legacy term "System Control Processor (SCP)" is sometimes used to refer to these firmware images.

- Firmware that runs in the highest privileged exception levels (EL3 and SEL1), which includes the Altra Max ARM Trusted Firmware (ATF) image.
- Some secure run-time application images that run in secure exception level 0 (SEL0) are considered to be SoC firmware, including the UEFI Management Mode (MM) image and TPM secure application images.
- These firmware images are platform-agnostic. The NVPARAM image, which is built as part of the platform firmware image, exposes platform customization hooks.

SoC firmware is delivered only as binaries.

1.14 Debug Support

Altra Max processors include hardware and software to support the Arm external debug mode. It also includes hardware trace blocks that support real-time tracing of instruction streams running in the Altra Max cores.

The processor is designed to the *ARMv8 Architecture Reference Manual, Revision F*, *Arm CoreSight Architecture Specification v3.0*, and *Arm CoreSight Base System Architecture 1.0, Combination B.*

For more information, refer to *Arm Debug Interface Architecture Specification ADIv6.0* for the PCP (Armv8) Debug Access Point (DAP), and to *Arm Debug Interface Architecture Specification ADIv5.0 to ADIv5.2* for the SMpro and PMpro DAPs.

For details, see *Chapter 13, "Debug and Trace."*

Processor Complex (PCP) 2

This chapter describes the Altra Max PCP. The chapter covers these topics:

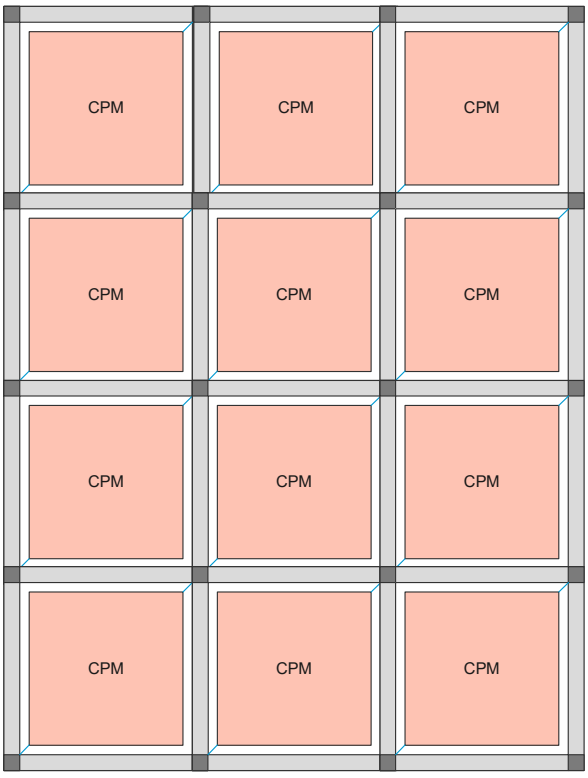
2.1 Overview

The PCP encompasses the coherence domain in which memory accesses are kept coherent. The PCP, shown in *Figure 2-1*, comprises:

- Up to 128 single-thread PEs, arranged in pairs in CPMs.
- The CMI, shown as a grey grid in *Figure 2-1*, connects the CPMs. The CMI contains a 16 MB SLC shared by the CPMs. The CMI also connects the PCP to the PCIe RCs and the system block.

Figure 2-1 provides a logical view of one quadrant of the PCP; the grey grid connects the blocks labeled "CPM." Each of the 12 CPMs in the quadrant contains two PEs. Refer to *Figure 1-1* to see how the PCP components work with other components, such as PCIe RCs and the system block.

Figure 2-1: PCP Block Diagram (Quadrant)



2.1.1 Cluster Processor Modules (CPMs)

The PCP contains up to 64 dual-core CPMs, for a total of up to 128 CPU cores. The Altra Max cores support the architecture described in the *Arm Architecture Reference Manual ARMv8*, including the 64-bit execution state (AArch64) and a 32-bit execution state (AArch32) that is compatible with previous versions of the Arm architecture.

In addition, each CPM features:

- 64 KB L1 i-caches, one per core.
- 64 KB L1 d-caches, one per core.
- 1 MB L2 unified caches, one per core.
- Hardware prefetch in L1 and L2 caches.
- Hardware cache coherency across all caches.
- Hardware virtualization support.
- Hardware tablewalk and nested page tables.
- End-to-end data poisoning and error isolation, for true server-class reliability.
- Static and dynamic power management features.

Section 2.2, "Cluster Processor Module (CPM)," provides more details.

2.1.2 Coherent Mesh Interconnect (CMI)

The CMI comprises a coherent mesh network containing:

- Various mesh nodes that connect mesh traffic to the CPUs, SLC, memory, and IO interfaces.
- 16 MB of SLC.
- Asynchronous FIFOs for interfacing with SoC components and MCUs.

Section 2.3, "Coherent Mesh Interconnect (CMI)," provides more details.

2.1.3 System Level Cache (SLC)

The PCP provides a 16 MB SLC shared by the CPMs and certain IO devices.

The SLC can store processor memory accesses and accesses from IO devices.

Section 2.4, "System Level Cache (SLC)," provides more details.

2.1.4 Memory Controller Units (MCUs)

An Altra Max processor provides eight MCUs that enable PCP and SoC agents to access external memory DIMMs. The MCUs implement logic to efficiently schedule DRAM accesses and to manage and maintain memory state.

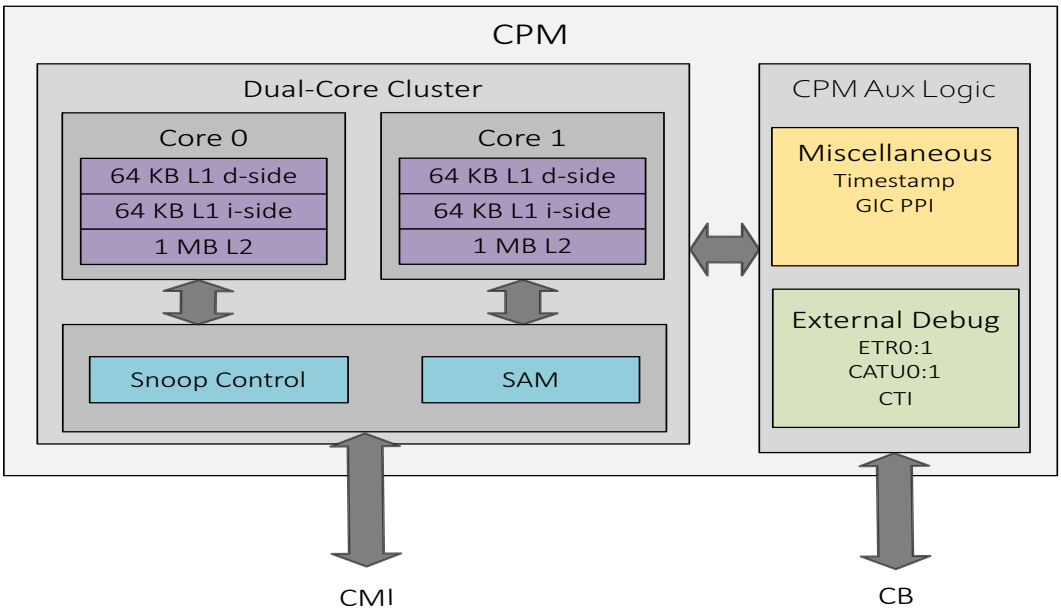
The MCUs are described in detail in *Chapter 2, "Processor Complex (PCP)."*

2.2 Cluster Processor Module (CPM)

The PCP contains up to 64 CPMs. *Figure 2-2* shows CPM components and interfaces. The main components are:

- Two cores.
  - Each core has a 64 KB L1 i-cache and 64 KB L1 d-cache.
  - Each core has a 1 MB L2 cache.
- A block containing snoop control logic and a System Address Map (SAM). A SAM, instantiated in each CPM, is associated with the CMI and facilitates accesses to the SLC, IO agents, and SoC devices.
- An auxiliary block containing miscellaneous and debug/trace logic.

Figure 2-2: CPM Block Diagram



2.2.1 Cluster Processor Module (CPM) Reliability, Availability, and Serviceability (RAS)

See *Section 8.5.1, "Altra Max Core Error Record Registers,"* for detailed information about RAS error records associated with the CPUs in the dual-CPU cluster, and *Section 8.5.2, "Snoop Control/System Address Map (SAM) Error Records,"* for detailed information about RAS error records associated with the snoop control/SAM block.

2.2.2 Virtualization

An Altra Max processor provides hardware acceleration for numerous active VMs. Each guest OS may or may not be aware of the underlying hypervisor layer. A guest OS running in a VM can access only specific regions of system memory, Virtual Address (VA) spaces, defined by the hypervisor. For each guest OS, VAs are mapped to Altra Max processor PAs. Each guest OS can access only the hardware resources assigned to it by the hypervisor, independently and in a way that does not interfere with

hardware resources assigned to other guest OSs. If a guest OS accesses the resources of another guest OS, the system detects the request and protects the other guest OSs. This protection does not impact other well-behaved guest OSs.

2.2.3 Cache Topology

An Altra Max processor provides three cache levels. Each of the 128 cores has an split L1 cache containing a 64 KB i-cache and a 64 KB d-cache, along with a unified 1 MB L2 cache.

An Altra Max processor does not have a traditional Last-Level Cache (LLC). Rather than a traditional Level 3 (L3) processor-side cache, Altra Max implements a 16 MB memory-side cache, called the SLC, that is shared among the CMN nodes. See *Section 2.4, "System Level Cache (SLC),"* for more information about the SLC.

2.2.4 Cache Coherency

Data cache coherency is maintained in instruction, data, and unified caches between multiple cores using the Modified, Exclusive, Shared, Invalid (MESI) protocol for L1 i-caches, L1 d-caches, and L2 caches.

The MESI protocol implements these cache states:

- Modified (M): The processor modifies the cache line in the current cache so that its value does not match the data in main memory; the cache line is therefore considered dirty. The cache line must be written back to memory before main memory content can be considered valid again.

  After the cache line is written back to memory, its status in the current cache changes to Shared (S).
- Exclusive (E): Only one core has the data in its cache, and the data is clean, that is, the data matches the data in main memory.

  The cache line may be changed to Shared (S) if another core reads the same main memory location.

  The cache line may be changed to Modified (M) if the processor writes to the cache line.
- Shared (S): Multiple cores have the data in their cache, and the data is clean, that is, the data matches the data in main memory.

  The cache line may be changed to Invalid (I) when it is discarded from the cache.
- Invalid (I): The cache line is currently not present in the cache.

2.3 Coherent Mesh Interconnect (CMI)

The CMI comprises a coherent mesh network containing:

- Various types of mesh nodes that connect mesh traffic to the CPUs, SLC, memory, and IO interfaces.
- 16 MB of SLC, described in *Section 2.4, "System Level Cache (SLC)."*

2.3.1 System Address Maps (SAMs)

All accesses require a target ID to route requests and data from source to destination. For addressable requests, the target ID is determined by a SAM that can generate an addressable request. Each CPM contains a SAM.

When a core or an primary IO device generates a request, the SAM lookup checks the defined address regions to determine the target ID of the request.

The SAM lookup first checks the GIC region, then the PCIe regions (and, in 2P systems, resources on the other socket) in priority order, and then checks DRAM regions local to the socket or System Cache Groups (SCGs).

2.3.2 Processor Access to Memory

The Altra Max cores access main memory through CMI components. For DRAM accesses, memory channels are optimized to minimize latency and maximize bandwidth and parallelism. See *Chapter 3, "Memory Controller Unit (MCU),"* for information about memory channels.

2.3.2.1 Non-Uniform Memory Access (NUMA)

The Altra Max processor supports these NUMA control modes, collectively referred to as Ampere NUMA Control (ANC):

- Monolithic.

  All cores in a socket have the same access to the eight available memory channels on the socket. The memory map presents a contiguous address space containing all memory installed in the DIMM slots.

- Hemisphere.

  In this mode, half of the cores in a socket are grouped in a node to which four memory channels on the socket are assigned; the remaining cores in the socket are assigned to another node along with the remaining four memory channels. The cores in each hemisphere are in one half of the physical SoC package and are associated with the closest MCUs.

  Because the cores and MCUs in each hemisphere are on average physically closer than they might be in a monolithic implementation, memory accesses have a lower latency than in a monolithic implementation. Accesses to the four memory channels in the other hemisphere have a higher latency.

  Although any core in one node can be made to access data in memory belonging to the other node, the OS typically has cores use memory belonging to the same node; the average lower memory latency results in better performance overall.

  The memory map for each hemisphere presents a contiguous address space containing all of the memory installed in the DIMM slots that are associated with the memory channels assigned to the node.

- Quadrant.

  This mode divides the cores in a socket into quadrants. Each quadrant contains a quarter of the cores of the processor along with the physically closest MCU pair.

  As in hemisphere mode, the physical proximity of cores and MCUs in the same node reduces the average latency of memory accesses within a quadrant. Accesses to memory channels in the other quadrants have higher latencies.

  The memory map for each quadrant presents a contiguous address space containing all of the memory installed in the DIMM slots associated with the memory channels assigned to the node.

2.3.3 Primary Input/Output (IO) Device Access to Memory

Primary IO devices access memory through dedicated addresses in the address map.

2.3.4 Processor Access to Secondary Input/Output (IO) Devices

The Altra Max cores access secondary IO devices through dedicated addresses in the SAMs.

2.3.5 Sample System Address Map (SAM) Configurations

Table 2-1 shows a sample SAM configuration for Socket 0 in a 1P configuration (primary socket only).

Table 2-1: Sample SAM Configuration for Socket 0: Primary Socket, 1P Configuration

DESCRIPTION	BASE ADDRESS	SIZE
GIC region	0x1000 0000 0000	128 KB
Socket 0 DRAM region	0x0000 0000 0000	8 TB
PCIe RcA3 64-bit MMIO + Translation Control Unit (TCU) + Control and Status Registers (CSRs)	0x3C00 0000 0000	4 TB
PCIe RcA3 32-bit MMIO	0x0000 7000 0000	256 MB
PCIe RcA2 64-bit MMIO + TCU + CSRs	0x3800 0000 0000	4 TB
PCIe RcA2 32-bit MMIO	0x0000 6000 0000	256 MB
PCIe RcA1 64-bit MMIO + TCU + CSRs	0x3400 0000 0000	4 TB
PCIe RcA1 32-bit MMIO	0x0000 5000 0000	256 MB
PCIe RcA0 64-bit MMIO + TCU + CSRs	0x3000 0000 0000	4 TB
PCIe RcA0 32-bit MMIO	0x0000 4000 0000	256 MB
PCIe RcA7 64-bit MMIO + TCU + CSRs	0x2C00 0000 0000	4 TB
PCIe RcA7 32-bit MMIO	0x0000 3000 0000	256 MB
PCIe RcA6 64-bit MMIO + TCU + CSRs	0x2800 0000 0000	4 TB
PCIe RcA6 32-bit MMIO	0x0000 2000 0000	256 MB
PCIe RcA5 64-bit MMIO + TCU + CSRs	0x2400 0000 0000	4 TB
PCIe RcA5 32-bit MMIO	0x0000 1000 0000	256 MB
PCIe RcA4 64-bit MMIO + TCU + CSRs	0x2000 0000 0000	4 TB
PCIe RcA4 32-bit MMIO	0x0000 0000 0000	256 MB
SYS complex	0x1000 0000 0000	4 GB
GIC region (GICR, GITS pages)	0x1000 0000 0000	16 MB

Table 2-2 shows a sample SAM configuration for Socket 0 in a 2P configuration (primary socket).

Table 2-2: Sample SAM Configuration for Socket 0: Primary Socket, 2P Configuration (Sheet 1 of 2)

DESCRIPTION	BASE ADDRESS	SIZE
GIC region	0x1000 0000 0000	128 KB
Socket 0 DRAM region	0x0000 0000 0000	8 TB
PCIe RcA3 64-bit MMIO + TCU + CSRs	0x3C00 0000 0000	4 TB
PCIe RcA3 32-bit MMIO	0x0000 3800 0000	128 MB
PCIe RcA2 64-bit MMIO + TCU + CSRs	0x3800 0000 0000	4 TB

Table 2-2: Sample SAM Configuration for Socket 0: Primary Socket, 2P Configuration (Sheet 2 of 2)

DESCRIPTION	BASE ADDRESS	SIZE
PCIe RcA2 32-bit MMIO	0x0000 3000 0000	128 MB
PCIe RcA1 64-bit MMIO + TCU + CSRs	0x3400 0000 0000	4 TB
PCIe RcA1 32-bit MMIO	0x0000 2800 0000	128 MB
PCIe RcA0 64-bit MMIO + TCU + CSRs	0x3000 0000 0000	4 TB
PCIe RcA0 32-bit MMIO	0x0000 2000 0000	128 MB
PCIe RcA7 64-bit MMIO + TCU + CSRs	0x2C00 0000 0000	4 TB
PCIe RcA7 32-bit MMIO	0x0000 1800 0000	128 MB
PCIe RcA6 64-bit MMIO + TCU + CSRs	0x2800 0000 0000	4 TB
PCIe RcA6 32-bit MMIO	0x0000 1000 0000	128 MB
PCIe RcA5 64-bit MMIO + TCU + CSRs	0x2400 0000 0000	4 TB
PCIe RcA5 32-bit MMIO	0x0000 0800 0000	128 MB
PCIe RcA4 64-bit MMIO + TCU + CSRs	0x2000 0000 0000	4 TB
PCIe RcA4 32-bit MMIO	0x0000 0000 0000	128 MB
SYS Complex	0x1000 0000 0000	4 GB
GIC region (GICR, GITS pages)	0x1000 0000 0000	16 MB
Define a CCIX node if one link is used for 2P port aggregation among two CCIX ports.	0x4000 0000 0000	64 TB
Define a CCIX node if one link is used for 2P port aggregation among two CCIX ports.	0x0000 C000 0000	1 GB
Define a CCIX node if one link is used for 2P port aggregation among two CCIX ports.	0x0000 4000 0000	1 GB

Table 2-3 shows a sample SAM configuration for Socket 1 in a 2P configuration (secondary socket).

Table 2-3: Sample SAM Configuration for Socket 1: Secondary Socket, 2P Configuration (Sheet 1 of 2)

DESCRIPTION	BASE ADDRESS	SIZE
GIC region	0x1000 0000 0000	128 KB
Socket 1 DRAM region	0x0000 0000 0000	128 TB
PCIe RcA3 64-bit MMIO + TCU + CSRs	0x7C00 0000 0000	4 TB
PCIe RcA3 32-bit MMIO	0x0000 B800 0000	128 MB
PCIe RcA2 64-bit MMIO + TCU + CSRs	0x7800 0000 0000	4 TB
PCIe RcA2 32-bit MMIO	0x0000 B000 0000	128 MB
PCIe RcA1 64-bit MMIO + TCU + CSRs	0x7400 0000 0000	4 TB
PCIe RcA1 32-bit MMIO	0x0000 A800 0000	128 MB
PCIe RcA0 64-bit MMIO + TCU + CSRs	0x7000 0000 0000	4 TB

Table 2-3: Sample SAM Configuration for Socket 1: Secondary Socket, 2P Configuration (Sheet 2 of 2)

DESCRIPTION	BASE ADDRESS	SIZE
PCIe RcA0 32-bit MMIO	0x0000 A000 0000	128 MB
PCIe RcA7 64-bit MMIO + TCU + CSRs	0x6C00 0000 0000	4 TB
PCIe RcA7 32-bit MMIO	0x0000 9800 0000	128 MB
PCIe RcA6 64-bit MMIO + TCU + CSRs	0x6800 0000 0000	4 TB
PCIe RcA6 32-bit MMIO	0x0000 9000 0000	128 MB
PCIe RcA5 64-bit MMIO + TCU + CSRs	0x6400 0000 0000	4 TB
PCIe RcA5 32-bit MMIO	0x0000 8800 0000	128 MB
PCIe RcA4 64-bit MMIO + TCU + CSRs	0x6000 0000 0000	4 TB
PCIe RcA4 32-bit MMIO	0x0000 8000 0000	128 MB
SYS complex	0x5000 0000 0000	4 GB
GIC region (GICR, GITS pages)	0x5000 0000 0000	16 MB
Define a CCIX node if one link for is used for 2P port aggregation among two or four CCIX ports.	0x2000 0000 0000	32 TB
Socket 0 SYS complex + Socket 0 GIC Interrupt Translation Service (ITS)	0x1000 0000 0000	8 GB
32-bit MMIO	0x0000 0000 0000	1 GB
DRAM	0x0000 8000 0000	1 GB
DRAM	0x0080 0000 0000	512 GB
DRAM	0x0100 0000 0000	1 TB
DRAM	0x0200 0000 0000	2 TB
DRAM	0x0400 0000 0000	512 GB

2.4 System Level Cache (SLC)

The SLC is a memory-side cache that is mostly exclusive with the L2 caches. The SLC is used for processor evictions and caches large data and instruction structures to improve system performance. The SLC is not a traditional processor-side Last Level Cache (LLC), sometimes called an L3 or L4 cache.

The 16 MB SLC is distributed among certain types of nodes in the CMI. Each of these nodes features a Combined Point of Coherency (PoC) and Point of Serialization (PoS). Distributing the SLC among these nodes provides improved performance when compared with a traditional L3 cache.

For 1P systems in monolithic NUMA mode, the SLC functions as an LLC that resembles a traditional 16 MB (L3) cache. For 2P systems in which both processors are in monolithic mode, TBD. For 1P and 2P systems In hemisphere and quadrant NUMA modes, and for 2P systems in monolithic mode, the SLC is distributed among subsets of the CMI nodes. In hemisphere mode, half of these CMI nodes are distributed in each hemisphere. In quadrant mode, a quarter of these CMI nodes are distributed in

each quadrant. See *Section 2.3.2.1, "Non-Uniform Memory Access (NUMA),"* for more information about supported NUMA nodes.

Along with memory accesses, the SLC can store accesses from the PCIe RCs.

2.5 Processor Core Registers

This section describes the processor core registers in the Altra Max processor:

- AArch32 registers by functional group.
- AArch64 registers by functional group.

2.5.1 AArch32 Registers by Functional Group

Table 2-4 identifies the architecturally defined registers that are implemented in the Altra Max processor.

Refer to the Armv8-A architecture profile in *Arm Architecture Reference Manual Armv8* for descriptions of these registers.

For the registers listed in *Table 2-4*, coproc==0b1111.

Table 2-4: AArch32 Identification Registers (Sheet 1 of 2)

NAME	TYPE	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
CNTFRQ	–	c14	0	c0	0	32	Timer clock ticks/second
CNTP_CTL	–	c14	0	c2	1	32	Counter-Timer Physical Timer Control
CNTP_CVAL	–	–	2	c14	–	64	Counter-Timer Physical Timer CompareValue
CNTP_TVAL	–	c14	0	c2	0	32	Counter-Timer Physical Timer TimerValue
CNTPCT	–	–	0	c14	–	64	Counter-Timer Physical Count
CNTV_CTL	–	c14	0	c3	1	32	Counter-Timer Virtual Timer control
CNTV_CVAL	–	–	3	c14	–	64	Counter-Timer Virtual Timer CompareValue
CNTV_TVAL	–	c14	0	c3	0	32	Counter-Timer Virtual Timer TimerValue
CNTVCT	–	–	1	c14	–	64	Counter-Timer Virtual Count
CP15ISB	–	c7	0	c5	4	32	Instruction Synchronization Barrier System Instruction
CP15DSB	–	c7	0	c10	4	32	Data Synchronization Barrier System Instruction
CP15DMB	–	c7	0	c10	5	32	Data Memory Barrier System Instruction
DLR	–	c4	3	c5	1	32	Debug Link Register
DSPSR	–	c4	3	c5	0	32	Debug Saved Program Status Register
TPIDRURO	–	c13	0	c0	3	32	User Read Only Thread ID

Table 2-4: AArch32 Identification Registers (Sheet 2 of 2)

NAME	TYPE	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
TPIDRURW	–	c13	0	c0	2	32	User Read/Write Thread ID Register

2.5.2 AArch64 Registers by Functional Group

This section describes by function architecturally defined registers that are implemented in the Altra Max processor.

Refer to the Armv8-A architecture profile in *Arm Architecture Reference Manual Armv8* for descriptions of these registers.

2.5.2.1 Identification Registers

Table 2-5: AArch64 Identification Registers (Sheet 1 of 3)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
AIDR_EL1	RO	3	c0	1	c0	7	32	Auxiliary ID Register EL1
CCSIDR_EL1	RO	3	c0	1	c0	2	32	Cache Size ID Register EL1
CLIDR_EL1	RO	3	c0	1	c0	1	64	Cache Level ID Register EL1
CSSELR_EL1	RW	3	c0	2	c0	0	32	Cache Size Selection Register EL1
CTR_EL0	RO	3	c0	3	c0	1	32	Cache Type Register EL1
DCZID_EL0	RO	3	c0	3	c0	7	32	Data Cache Zero ID Register EL0
ERRIDR_EL1	RO	3	c5	0	c3	0	32	Error ID Register EL1
ID_AA64AFR0_EL1	RO	3	c0	0	c5	4	64	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	RO	3	c0	0	c5	5	64	AArch64 Auxiliary Feature Register 1
ID_AA64DFR0_EL1	RO	3	c0	0	c5	1	64	AArch64 Debug Feature Register 0
ID_AA64PFR1_EL1	RO	3	c0	0	c4	1	64	AArch64 Core Feature Register 1
ID_AA64ISAR0_EL1	RO	3	c0	0	c6	0	64	AArch64 Instruction Set Attribute Register 0 EL1
ID_AA64ISAR1_EL1	RO	3	c0	0	c6	1	64	AArch64 Instruction Set Attribute Register 1 EL1
ID_AA64MMFR0_EL1	RO	3	c0	0	c7	0	64	AArch64 Memory Model Feature Register 0 EL1
ID_AA64MMFR1_EL1	RO	3	c0	0	c7	1	64	AArch64 Memory Model Feature Register 1 EL1
ID_AA64MMFR2_EL1	RO	3	c0	0	c7	2	64	AArch64 Memory Model Feature Register 2 EL1
ID_AA64PFR0_EL1	RO	3	co	0	c4	0	64	AArch64 Processor Feature Register 0 EL1

Table 2-5: AArch64 Identification Registers (Sheet 2 of 3)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
ID_AA64PFR1_EL1	RO	3	c0	0	c4	1	64	AArch64 Core Feature Register 1 EL1
ID_AFR0_EL1	RO	3	c0	0	c1	3	32	AArch32 Auxiliary Feature Register 0 EL1
ID_DFR0_EL1	RO	3	c0	0	c1	2	32	AArch32 Debug Feature Register 0 EL1
ID_ISAR0_EL1	RO	3	c0	0	c2	0	32	AArch32 Instruction Set Attribute Register 0 EL1
ID_ISAR1_EL1	RO	3	c0	0	c2	1	32	AArch32 Instruction Set Attribute Register 1 EL1
ID_ISAR2_EL1	RO	3	c0	0	c2	2	32	AArch32 Instruction Set Attribute Register 2 EL1
ID_ISAR3_EL1	RO	3	c0	0	c2	3	32	AArch32 Instruction Set Attribute Register 3 EL1
ID_ISAR4_EL1	RO	3	c0	0	c2	4	32	AArch32 Instruction Set Attribute Register 4 EL1
ID_ISAR5_EL1	RO	3	c0	0	c2	5	32	AArch32 Instruction Set Attribute Register 5 EL1
ID_ISAR6_EL1	RO	3	c0	0	c2	7	32	AArch32 Instruction Set Attribute Register 6 EL1
ID_MMFR0_EL1	RO	3	c0	0	c1	4	32	AArch32 Memory Model Feature Register 0 EL1
ID_MMFR1_EL1	RO	3	c0	0	c1	5	32	AArch32 Memory Model Feature Register 1 EL1
ID_MMFR2_EL1	RO	3	c0	0	c1	6	32	AArch32 Memory Model Feature Register 2 EL1
ID_MMFR3_EL1	RO	3	c0	0	c1	7	32	AArch32 Memory Model Feature Register 2 EL1
ID_MMFR4_EL1	RO	3	c0	0	c2	6	32	AArch32 Memory Model Feature Register 4 EL1
ID_PFR0_EL1	RO	3	c0	0	c1	0	32	AArch32 Processor Feature Register 0 EL1
ID_PFR1_EL1	RO	3	c0	0	c1	1	32	AArch32 Processor Feature Register 1 EL1
ID_PFR2_EL1	RO	3	c0	0	c3	4	32	AArch32 Processor Feature Register 2 EL1
LORID_EL1	RO	3	c10	0	c4	7	64	LORegion ID Register EL1
MIDR_EL1	RO	3	c0	0	c0	0	64	Main ID Register EL1
MPIDR_EL1	RO	3	c0	0	c0	5	64	Multiprocessor Affinity Register EL1

Table 2-5: AArch64 Identification Registers (Sheet 3 of 3)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
REVIDR_EL1	RO	3	c0	0	c0	6	32	Revision ID Register EL1
VMPIDR_EL2	RW	3	c0	4	c0	5	64	Virtualization Multiprocessor ID Register EL2
VPIDR_EL2	RW	3	c0	4	c0	0	32	Virtualization Core ID Register EL2

2.5.2.2 System Control Registers

Table 2-6: System Control Registers

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
ACTLR_EL1	RW	3	c1	0	c0	1	32	Auxiliary Control Register EL1
ACTLR_EL2	RW	3	c1	4	c0	1	64	Auxiliary Control Register EL2
ACTLR_EL3	RW	3	c2	6	c0	1	64	Auxiliary Control Register EL3
CPACR_EL1	RW	3	c1	5	c0	2	32	Architectural Feature Access Control Register
SCTLR_EL1	RW	3	c1	0	c0	0	32	System Control Register EL1
SCTLR_EL2	RW	3	c1	4	c0	0	32	System Control Register EL2
SCTLR_EL3	RW	3	c1	6	c0	0	32	System Control Register EL3
SCTLR_EL12	RW	3	c1	5	c0	0	32	System Control Register EL12

2.5.2.3 Reliability, Availability, and Serviceability (RAS) Registers

For information about additional processor core RAS registers, see *"Altra Max Core Error Record Registers" on page 8-15*.

Table 2-7: RAS Registers (Sheet 1 of 2)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
DISR_EL1	RW	3	c12	0	c1	1	64	Deferred Interrupt Status Register EL1
ERRIDR_EL1	RW	3	c5	0	c3	0	32	Error ID Register EL1
ERRSELR_EL1	RW	3	c5	0	c3	1	32	Error Record Select Register EL1
ERXADDR_EL1	RW	3	c5	0	c4	3	64	Selected Error Record Address Register EL1
ERXCTLR_EL1	RW	3	c5	0	c4	1	64	Selected Error Record Control Register EL1
ERXFR_EL1	RW	3	c5	0	c4	0	64	Selected Error Record Feature Register EL1
ERXMISC0_EL1	RW	3	c5	0	c5	0	64	Selected Error Record Miscellaneous Register 0 EL1

Table 2-7: RAS Registers (Sheet 2 of 2)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
ERXMISC1_EL1	RW	3	c5	0	c5	1	64	Selected Error Record Miscellaneous Register 0 EL1
ERXSTATUS_EL1	RW	3	c5	0	c4	2	32	Selected Error Record Primary Status Register EL1
ERXPFGCDNR_EL1	RW	3	c15	0	c2	2	32	Selected Error Pseudo Fault Generation Countdown Register EL1
ERXPFCTLR_EL1	RW	3	c15	0	c2	1	32	Selected Error Pseudo Fault Generation Control Register EL1
ERXPFGFR_EL1	RO	3	c15	0	c2	0	32	Selected Error Pseudo Fault Generation Feature Register EL1
HCR_EL2	RW	3	c1	4	c1	0	64	Hypervisor Configuration Register EL2
VDISR_EL2	RW	3	c12	4	c1	1	64	Virtual Deferred Interrupt Status Register EL2
VSESR_EL2	RW	3	c5	4	c2	3	64	Virtual SError Exception Syndrome Register EL2

2.5.2.4 Virtual Memory Control Registers

Table 2-8: Virtual Memory Control Registers (Sheet 1 of 2)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
AMAIR_EL1	RW	3	c10	0	c3	0	64	Auxiliary Memory Attribute Indirection Register EL1
AMAIR_EL2	RW	3	c10	4	c3	0	64	Auxiliary Memory Attribute Indirection Register EL2
AMAIR_EL3	RW	3	c10	6	c3	0	64	Auxiliary Memory Attribute Indirection Register EL3
LORC_EL1	RW	3	c10	0	c4	3	64	LORegion Control Register EL1
LOREA_EL1	RW	3	c10	0	c4	1	64	LORegion End Address Register EL1
LORID_EL1	R0	3	c10	0	c4	7	64	LORegion ID Register EL1
LORN_EL1	RW	3	c10	0	c4	2	64	LORegion Number Register EL1
LORSA_EL1	RW	3	c10	0	c4	0	64	LORegion Start Address Register EL1
TCR_EL1	RW	3	c2	0	c0	2	64	Translation Control Register EL1
TCR_EL2	RW	3	c2	4	c0	2	64	Translation Control Register EL2
TCR_EL3	RW	3	c2	6	c0	2	64	Translation Control Register EL3
TTBR0_EL1	RW	3	c2	0	c0	0	64	Translation Table Base Register 0 EL1
TTBR0_EL2	RW	3	c2	4	c0	0	64	Translation Table Base Register 0 EL2

Table 2-8: Virtual Memory Control Registers (Sheet 2 of 2)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
TTBR0_EL3	RW	3	c2	6	c0	0	64	Translation Table Base Register 0 EL3
TTBR1_EL1	RW	3	c2	0	c0	1	64	Translation Table Base Register 1 EL1
TTBR1_EL2	RW	3	c2	4	c0	1	64	Translation Table Base Register 1 EL2
VTTBR_EL2	RW	3	c2	4	c1	0	64	Virtualization Translation Table Base Register EL2

2.5.2.5 Virtualization Registers

Table 2-9: Virtualization Registers

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
ACTLR_EL2	RW	3	c1	4	c0	1	64	Auxiliary Control Register EL2
AFSR0_EL2	RW	3	c5	4	c1	0	32	Auxiliary Fault Status Register 0 EL2
AFSR1_EL2	RW	3	c5	4	c1	0	32	Auxiliary Fault Status Register 0 EL2
AMAIR_EL2	RW	3	c10	4	c3	0	64	Auxiliary Memory Attribute Indirection Register EL2
CPTR_EL2	RW	3	c1	4	c1	2	32	Architectural Feature Trap Register EL2
ESR_EL2	RW	3	c5	4	c2	0	32	Exception Syndrome Register EL2
HACR_EL2	RW	3	c1	4	c1	7	32	Hypervisor Auxiliary Configuration Register EL2
HCR_EL2	RW	3	c10	0	c4	7	64	Hypervisor Configuration Register EL2
HPFAR_EL2	RW	3	c6	4	c0	4	64	Hypervisor IPA Fault Address Register EL2
TCR_EL2	RW	3	c2	4	c0	2	64	Translation Control Register EL2
VMPIDR_EL2	RW	3	c0	4	c0	5	64	Virtualization Multiprocessor ID Register EL2
VPIDR_EL2	RW	3	c0	4	c0	0	32	Virtualization Core ID Register EL2
VSESR_EL2	RW	3	c5	4	c2	3	64	Virtual SError Exception Syndrome Register EL2
VTCR_EL2	RW	3	c2	4	c1	2	32	Translation Table Base Register 0 EL3
VTTBR_EL2	RW	3	c2	4	c1	0	64	Virtualization Translation Table Base Register EL2

2.5.2.6 Exception and Fault Handling Registers

Table 2-10: Exception and Fault Handling Registers

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
AFSR0_EL1	RW	3	c5	0	c1	0	64	Auxiliary Fault Status Register 0 EL1
AFSR0_EL2	RW	3	c5	4	c1	0	32	Auxiliary Fault Status Register 0 EL2
AFSR0_EL3	RW	3	c5	6	c1	0	32	Auxiliary Fault Status Register 0 EL2
AFSR1_EL1	RW	3	c5	0	c1	1	32	Auxiliary Fault Status Register 1 EL0
AFSR1_EL2	RW	3	c5	4	c1	1	32	Auxiliary Fault Status Register 1 EL2
AFSR1_EL3	RW	3	c5	6	c1	1	32	Auxiliary Fault Status Register 1 EL3
DISR_EL1	RW	3	c12	0	c1	1	64	Deferred Interrupt Status Register EL1
ESR_EL1	RW	3	c5	0	c2	0	32	Exception Syndrome Register EL1
ESR_EL2	RW	3	c5	4	c2	0	32	Exception Syndrome Register EL2
ESR_EL3	RW	3	c5	6	c2	0	32	Exception Syndrome Register EL3
HPFAR_EL2	RW	3	c6	4	c0	4	64	Hypervisor IPA Fault Address Register EL2
VDISR_EL2	RW	3	c12	4	c1	1	64	Virtual Deferred Interrupt Status Register EL2
VSESR_EL2	RW	3	c5	4	c2	3	64	Virtual SError Exception Syndrome Register EL2
VPIDR_EL2	RW	3	c0	4	c0	0	32	Virtualization Core ID Register EL2
VSESR_EL2	RW	3	c5	4	c2	3	64	Virtual SError Exception Syndrome Register EL2

2.5.2.7 Implementation Defined Registers

Table 2-11: Implementation Defined Registers

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
ERXPFGCDNR_EL1	RW	3	c15	0	c2	2	32	Selected Error Pseudo Fault Generation Countdown Register EL1
ERXPFCTLR_EL1	RW	3	c15	0	c2	1	32	Selected Error Pseudo Fault Generation Control Register EL1
ERXPFGFR_EL1	RO	3	c15	0	c2	0	32	Selected Error Pseudo Fault Generation Feature Register EL1

2.5.2.8 Security Registers

Table 2-12: Security Registers

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
ACTLR_EL3	RW	3	c2	6	c0	1	64	Auxiliary Control Register EL3
AFSR0_EL3	RW	3	c5	6	c1	0	32	Auxiliary Fault Status Register 0 EL3
AFSR1_EL3	RW	3	c5	6	c1	1	32	Auxiliary Fault Status Register 1 EL3
AMAIR_EL2	RW	3	c10	4	c3	0	64	Auxiliary Memory Attribute Indirection Register EL2
CPTR_EL2	RW	3	c1	4	c1	2	32	Architectural Feature Trap Register EL2
MDCR_EL3	RW	3	c1	6	c3	1	32	Monitor Debug Configuration Register EL3

2.5.2.9 Reset Management Registers

Table 2-13: Reset Management Registers

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
RMR_EL3	RW	3	c12	6	c0	2	32	Reset Management Register EL3
RVBAR_EL3	RW	3	c12	6	c0	1	64	Reset Vector Base Address Register EL3

2.5.2.10 Address Registers

Table 2-14: Address Registers

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
PAR_EL1	RW	3	c7	0	c4	0	64	Physical Address Register EL1

2.5.2.11 Miscellaneous Registers

Table 2-15: Miscellaneous Registers (Sheet 1 of 4)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
IFSR32_EL2	–	3	c5	4	c0	1	32	Instruction Halt Status Register
CNTFRQ_EL0	–	3	c14	3	c0	0	32	Counter-Timer Frequency Register
CNTHP_CTL_EL2	–	3	c14	4	c2	1	32	Counter-Timer Hypervisor Physical Timer Control Register
CNTHP_CVAL_EL2	–	3	c14	4	c2	2	64	Counter-Timer Hypervisor Physical CompareValue Register

Table 2-15: Miscellaneous Registers (Sheet 2 of 4)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
CNTHP_TVAL_EL2	–	3	c14	4	c2	0	32	Counter-Timer Hypervisor Physical Timer TimerValue Register
CNTHV_CTL_EL2	–	3	c14	4	c3	1	32	Counter-Timer Virtual Timer Control Register
CNTHV_CVAL_EL2	–	3	c14	4	c3	2	64	Counter-Timer Virtual Timer CompareValue Register
CNTHV_TVAL_EL2	–	3	c14	4	c3	0	32	Counter-timer Virtual Timer TimerValue register
CNTKCTL_EL1	–	3	c14	0	c1	0	32	Counter-Timer Kernel Control Register EL1
CNTKCTL_EL12	–	3	c14	5	c1	0	32	Counter-Timer Kernel Control Register EL2
CNTP_CTL_EL0	–	3	c14	3	c2	1	32	Counter-Timer Physical Timer Control Register EL0
CNTP_CTL_EL02	–	3	c14	5	c2	1	32	Counter-Timer Physical Timer Control Register EL02
CNTP_CVAL_EL0	–	3	c14	3	c2	1	32	Counter-Timer Physical Timer CompareValue Register EL0
CNTP_CTL_EL0	–	3	c14	3	c2	1	32	Counter-Timer Physical Timer Control Register EL0
CNTP_CTL_EL02	–	3	c14	5	c2	1	32	Counter-Timer Physical Timer Control Register EL02
CNTP_CVAL_EL0	–	3	c14	3	c2	2	64	Counter-Timer Physical Timer CompareValue Register EL0
CNTP_CVAL_EL02	–	3	c14	5	c2	2	64	Counter-Timer Physical Timer CompareValue Register EL02
CNTP_TVAL_EL0	–	3	c14	3	c2	0	32	Counter-Timer Physical Timer TimerValue Register EL0
CNTP_TVAL_EL02	–	3	c14	5	c2	0	32	Counter-Timer Physical Timer TimerValue Register EL02
CNTPCT_EL0	–	3	c14	3	c0	1	64	Counter-Timer Physical Count Register EL0
CNTPS_CTL_EL1	–	3	c14	7	c2	1	32	Counter-Timer Physical Secure Timer Control Register EL1
CNTPS_CVAL_EL1	–	3	c14	7	c2	2	64	Counter-Timer Physical Secure Timer CompareValue Register EL1
CNTPS_TVAL_EL1	–	3	c14	7	c2	0	32	Counter-Timer Physical Secure Timer TimerValue Register EL1

Table 2-15: Miscellaneous Registers (Sheet 3 of 4)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
CNTV_CTL_EL0	–	3	c14	3	c3	1	32	Counter-Timer Virtual Timer Control Register EL0
CNTV_CTL_EL02	–	3	c14	5	c3	1	32	Counter-Timer Virtual Timer Control Register EL02
CNTV_CVAL_EL0	–	3	c14	3	c3	2	64	Counter-Timer Virtual Timer CompareValue Register EL0
CNTV_CVAL_EL02	–	3	c14	5	c3	2	64	Counter-Timer Virtual Timer CompareValue Register EL0
CNTV_TVAL_EL0	–	3	c14	3	c3	0	32	Counter-Timer Virtual Timer TimerValue Register EL0
CNTV_TVAL_EL02	–	3	c14	5	c3	0	32	Counter-Timer Virtual Timer TimerValue Register EL02
CNTVCT_EL0	–	3	c14	3	c0	2	64	Counter-timer Virtual Count register EL0
CNTVOFF_EL2	–	3	c14	4	c0	1	32	Counter-timer Virtual Count register EL2
CONTEXTIDR_EL1	–	3	c13	0	c0	1	32	Context ID Register EL1
CONTEXTIDR_EL2	–	3	c13	4	c0	1	32	Context ID Register EL2
CONTEXTIDR_EL12	–	3	c13	5	c0	1	32	Context ID Register EL12
CPACR_EL12	–	3	c1	5	c0	2	32	Architectural Feature Access Control Register
CPTR_EL3	–	3	c1	6	c1	2	32	Architectural Feature Trap Register EL3
DACR32_EL2	–	3	c3	4	c0	0	32	Domain Access Control Register EL2
ESR_EL12	–	3	c5	5	c2	0	32	Exception Syndrome RegisterEL12
FAR_EL1	–	3	c6	0	c0	0	64	Fault Address Register EL1
FAR_EL2	–	3	c6	4	c0	0	64	Fault Address Register EL2
FAR_EL3	–	3	c6	6	c0	0	64	Fault Address Register EL3
FAR_EL12	–	3	c6	5	c0	0	64	Fault Address Register EL12
FPEXC32_EL2	–	3	c5	4	c3	0	32	Floating-Point Exception Control Register
HSTR_EL2	–	3	c1	4	c1	3	32	Hypervisor System Trap Register EL2
ID_AA64DFR1_EL1	–	3	c0	0	c5	1	64	AArch64 Debug Feature Register 1 EL1
ID_AA64PFR1_EL1	–	3	c0	0	c4	1	64	AArch64 Core Feature Register 1 EL1
ISR_EL1	–	3	c12	0	c4	1	32	Interrupt Status Register ELI

Table 2-15: Miscellaneous Registers (Sheet 4 of 4)

NAME	TYPE	Opcd0	CRn	Opcd1	CRm	Opcd2	WIDTH	DESCRIPTION
MAIR_EL1	–	3	c10	0	c2	0	64	Memory Attribute Indirection Register EL1
MAIR_EL2	–	3	c10	4	c2	0	64	Memory Attribute Indirection Register EL2
MAIR_EL3	–	3	c10	6	c2	0	64	Memory Attribute Indirection Register EL3
MAIR_EL12	–	3	c10	5	c2	0	64	Memory Attribute Indirection Register EL12
MDCR_EL2	–	3	c1	4	c1	1	32	Monitor Debug Configuration Register EL2
MVFR0_EL1	–	3	c0	0	c3	0	32	AArch32 Media and VFP Feature Register 0 EL1
MVFR1_EL1	–	3	c0	0	3	1	32	AArch32 Media and VFP Feature Register 1 EL1
MVFR2_EL1	–	3	c0	0	c3	2	32	AArch32 Media and VFP Feature Register 2 EL1
SCR_EL3	–	3	c1	6	c1	0	32	Secure Configuration Register EL3
SDER32_EL3	–	3	c1	6	c1	1	32	AArch32 Secure Debug Enable Register EL3
TCR_EL12	–	3	c2	5	c0	2	64	Translation Control Register EL12
TPIDR_EL0	–	3	c13	3	c0	2	64	Read/Write Software Thread ID Register EL0
TPIDR_EL1	–	3	c13	0	c0	4	64	Software Thread ID Register EL1
TPIDR_EL2	–	3	c13	4	c0	2	64	Software Thread ID Register EL2
TPIDR_EL3	–	3	c13	6	c0	2	64	Software Thread ID Register EL3
TPIDRRO_EL0	–	3	c13	3	c0	3	64	Read-Only Software Thread ID Register EL0
TTBR0_EL12	–	3	c2	5	c0	0	64	Translation Table Base Register 0 EL12
TTBR1_EL12	–	3	c2	5	c0	1	64	Translation Table Base Register 1 EL12
VBAR_EL1	–	3	c12	0	c0	0	64	Vector Base Address Register EL1
VBAR_EL2	–	3	c12	4	c0	0	64	Vector Base Address Register EL2
VBAR_EL3	–	3	c12	6	c0	0	64	Vector Base Address Register EL3
VBAR_EL12	–	3	c12	5	c0	0	64	Vector Base Address Register EL12

Memory Controller Unit (MCU)

3

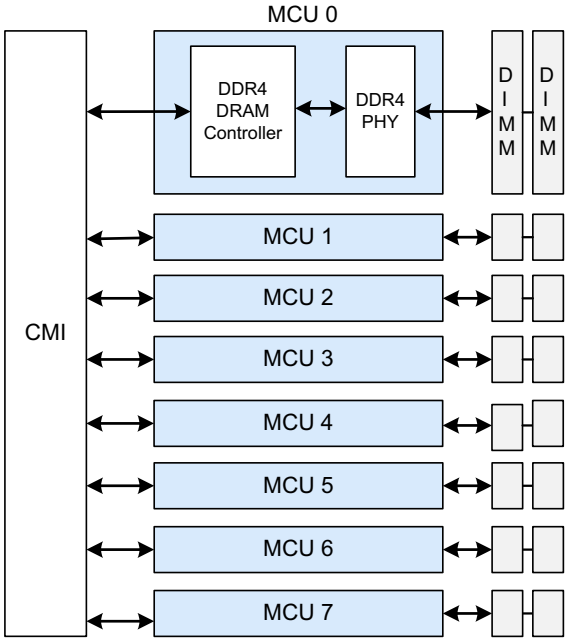
This chapter describes the MCU in the Altra Max processor. The chapter contains these topics:

3.1 Overview

An Altra Max processor provides eight MCUs that enable PCP and SoC agents to access external memory DIMMs. The MCUs implement logic to efficiently schedule DRAM accesses and to manage and maintain memory state.

Figure 3-1 shows the MCUs with other major SoC components.

Figure 3-1: MCU Block Diagram



Each MCU comprises a high-performance DDR4 controller and the corresponding physical layer (PHY) logic to control and access one or two DIMMs. The controller includes a deep request queue that supports efficient memory request scheduling. This optimizes DDR channel utilization and memory request latency.

3.2 Features

The MCU provides these features:

- Arm architecture security domains.
- Low-power operation through programmable DRAM power modes controlled by Ampere Computing SoC firmware.
- SECDED ECC mode.
- Command/address link protection for DDR4 link errors.
- CRC protection for write data transferred to DDR4 devices.
- Automated DRAM scrubbing.
- Performance profiling using event signals and counters.
- Interrupts for error and alert conditions.

3.3 Supported DDR4 Module Configurations

The MCU supports these DDR4 module configurations:

- Load-Reduced DIMM (LRDIMM).
- Registered DIMM (RDIMM).
- Unregistered DIMM (UDIMM).
- Stacked (3DS) RDIMM.

Refer to *Altra Max Platform Hardware Design Specification* for detailed information about supported DDR4 module configurations.

3.4 Supported Transfer Rates

The MCUs support these operating transfer rates:

- 1600 megatransfers per second (MT/s) (CLK = 800 MHz).
- 1866 MT/s (CLK = 933 MHz).
- 2133 MT/s (CLK = 1066 MHz).
- 2400 MT/s (CLK = 1200 MHz).
- 2667 MT/s (CLK = 1333 MHz).
- 2933 MT/s (CLK = 1467 MHz).
- 3200 MT/s (CLK = 1600 MHz).

3.5 Physical Layer (PHY) Configuration

The MCU PHY supports these parameters:

- A memory data width of 72 bits.
- A maximum frequency of 1600 MHz (DDR4-3200).
- Low-power states controlled by the DRAM controller.
- Up to four Chip Select (CS) outputs.
- Per-physical-rank leveling with up to four delay value sets.
- x4 and x8 devices.

3.6 Supported Memory Channel Configurations

An Altra Max processor supports eight memory channels providing up to 4 TB of memory per socket. The processor also supports six, four, two, and one memory controller configurations, but only those shown in *Table 3-1*. The single memory controller configuration is for debugging. Each channel provides DDR connections for two DIMM memory modules (2DPC).

Each MCU implements the interface and control logic for one DDR4 channel. This section describes requirements for supported memory channel configurations, included supported channel configurations and limits on the number of variations of the configured channels.

Refer to *Altra Max Platform Hardware Design Specification* for detailed information about supported memory channel configurations.

In *Table 3-1*, a "Y" indicates that the associated memory channel can be populated.

Table 3-1: Supported Memory Channel Configurations

NUMBER OF CHANNELS	CHANNELS USED							
	MCU0	MCU1	MCU2	MCU3	MCU4	MCU5	MCU6	MCU7
1	Y	—	—	—	—	—	—	—
1	—	—	—	—	Y	—	—	—
2	Y	—	—	—	Y	—	—	—
4	Y	Y	—	—	Y	Y	—	—
6	Y	Y	Y	—	Y	Y	Y	—
8	Y	Y	Y	Y	Y	Y	Y	Y

In a 1P system, channel configurations are supported for one, two, four, six, and eight channels. For 2P systems, each socket must comply with the installations shown in *Table 3-1*, but memory on the Socket 0 channels is not required to be the same size as the memory on the Socket 0 Channels. However, all channels on a socket must be identically sized.

3.6.1 Address Ranges for 1P Systems

Table 3-2: Address Ranges for 1P Systems (Sheet 1 of 2)

MEMORY SIZE (GB)	REGION DECODE RANGE	PHYSICAL MEMORY REGIONS	
		REGION 0	REGION 1
4	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x0080 FFFF FFFF
8	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x0081 FFFF FFFF
16	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x0083 FFFF FFFF
24	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x0085 FFFF FFFF
32	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x0087 FFFF FFFF
48	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x008B FFFF FFFF
64	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x008F FFFF FFFF
96	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x0097 FFFF FFFF
128	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x009F FFFF FFFF
192	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x00AF FFFF FFFF
256	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x00BF FFFF FFFF
384	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x00DF FFFF FFFF
512	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x00FF FFFF FFFF
768	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x013F FFFF FFFF
1024	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x017F FFFF FFFF
1536	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x01FF FFFF FFFF
2048	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x027F FFFF FFFF

Table 3-2: Address Ranges for 1P Systems (Sheet 2 of 2)

MEMORY SIZE (GB)	REGION DECODE RANGE	PHYSICAL MEMORY REGIONS	
		REGION 0	REGION 1
3072	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x037F FFFF FFFF
4096	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 FFFF FFFF	0x0080 8000 0000 — 0x047F FFFF FFFF

3.6.2 Address Ranges for Socket 0 in 2P Systems

Table 3-3: Address Ranges for Socket 0 in 2P Systems (Sheet 1 of 2)

SOCKET 0 MEMORY SIZE (GB)	REGION DECODE RANGE	PHYSICAL MEMORY REGIONS		
		REGION 0	REGION 1	REGION 2
4	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x0080 FFFF FFFF	0x0081 4000 0000 — 0x0081 7FFF FFFF
8	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x0081 FFFF FFFF	0x0082 4000 0000 — 0x0082 7FFF FFFF
16	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x0083 FFFF FFFF	0x0084 4000 0000 — 0x0084 7FFF FFFF
24	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x0085 FFFF FFFF	0x0086 4000 0000 — 0x0086 7FFF FFFF
32	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x0087 FFFF FFFF	0x0088 4000 0000 — 0x0088 7FFF FFFF
48	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x008B FFFF FFFF	0x008C 4000 0000 — 0x008C 7FFF FFFF
64	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x008F FFFF FFFF	0x0090 4000 0000 — 0x0090 7FFF FFFF
96	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x0097 FFFF FFFF	0x0098 4000 0000 — 0x0098 7FFF FFFF
128	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x009F FFFF FFFF	0x00A0 4000 0000 — 0x00A0 7FFF FFFF
192	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x00AF FFFF FFFF	0x00B0 4000 0000 — 0x00B0 7FFF FFFF
256	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x00BF FFFF FFFF	0x00C0 4000 0000 — 0x00C0 7FFF FFFF
384	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x00DF FFFF FFFF	0x00E0 4000 0000 — 0x00E0 7FFF FFFF
512	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x00FF FFFF FFFF	0x0100 4000 0000 — 0x0100 7FFF FFFF
768	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x013F FFFF FFFF	0x0140 4000 0000 — 0x0140 7FFF FFFF
1024	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x017F FFFF FFFF	0x0180 4000 0000 — 0x0180 7FFF FFFF
1536	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x01FF FFFF FFFF	0x0200 4000 0000 — 0x0200 7FFF FFFF
2048	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x027F FFFF FFFF	0x0280 4000 0000 — 0x0280 7FFF FFFF

Table 3-3: Address Ranges for Socket 0 in 2P Systems (Sheet 2 of 2)

SOCKET 0 MEMORY SIZE (GB)	REGION DECODE RANGE	PHYSICAL MEMORY REGIONS		
		REGION 0	REGION 1	REGION 2
3072	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x037F FFFF FFFF	0x0380 4000 0000 — 0x0380 7FFF FFFF
4096	0x0000 0000 0000 — 0x07FF FFFF FFFF	0x0000 8000 0000 — 0x0000 BFFF FFFF	0x0080 8000 0000 — 0x047F FFFF FFFF	0x0480 4000 0000 — 0x0480 7FFF FFFF

3.6.3 Address Ranges for Socket 1 in 2P Systems

Table 3-4: Address Ranges for Socket 1 in 2P Systems (Sheet 1 of 2)

SOCKET 1 MEMORY SIZE (GB)	REGION DECODE RANGE	PHYSICAL MEMORY REGIONS	
		REGION 0	REGION 1
4	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4081 3FFF FFFF
8	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4082 3FFF FFFF
16	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4084 3FFF FFFF
32	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4088 3FFF FFFF
24	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4086 3FFF FFFF
48	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x408C 3FFF FFFF
64	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4090 3FFF FFFF
96	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4098 3FFF FFFF
128	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x40A0 3FFF FFFF
192	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x40B0 3FFF FFFF
256	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x40C0 3FFF FFFF
384	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x40E0 3FFF FFFF
512	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4100 3FFF FFFF
768	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4140 3FFF FFFF
1024	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4180 3FFF FFFF
1536	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4200 3FFF FFFF
2048	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4280 3FFF FFFF

Table 3-4: Address Ranges for Socket 1 in 2P Systems (Sheet 2 of 2)

SOCKET 1 MEMORY SIZE (GB)	REGION DECODE RANGE	PHYSICAL MEMORY REGIONS	
		REGION 0	REGION 1
3072	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x4380 3FFF FFFF
4096	0x0000 0000 0000 — 0x7FFF FFFF FFFF	0x0000 C000 0000 — 0x0000 FFFF FFFF	0x4080 8000 0000 — 0x43FF FFFF FFFF

3.7 Security Domains

The MCU supports the Arm architectural security domains. DDR driver software configures the controller with access permission controls for up to eight protection regions plus the background region. Memory requests are transmitted to the MCU with an accompanying flag indicating whether the request originated from a secure world or normal world process. The MCU applies appropriate configured access permissions to each memory access; the corresponding flag determines which access permissions apply to the request.

3.8 DRAM Error Correction Code (ECC) Modes and Protection

In addition to a traditional SECDED ECC mode, the MCU supports a symbol-based ECC mode that protects against multiple errors. The symbol size is eight bits and is configurable to provide protection against transient or permanent nibble or byte lane failure. This feature can provide correction capability for up to four random symbol failures in a 64-byte burst, and subsequently eight systematic failures in a burst. This protects against:

- Random multibit failures in the DRAM array.
- Transient byte-lane transmission errors.
- The known failure of a single x8 DRAM chip in a rank.
- The known failure of two x4 DRAM chips in a rank.
- The random failure of a single x4 DRAM chip in a rank.

The symbol protection mode cannot be guaranteed to correct all such failures. However, the MCU does not perpetuate uncorrected bad data or falsely corrected bad data that it detects. The MCU marks all uncorrectable or improperly corrected data as corrupted, using poison indicators.

3.8.1 Command/Address Link Parity Protection

The MCUs support Command/Address (CA) parity when RDIMMs implementing CA parity checking are connected to a channel. This feature can detect transient command and address single-bit errors and initiate retries of failing operations.

3.8.2 Write Data Link Cyclic Redundancy Check (CRC) Protection

The MCU supports DDR4 write data CRC generation and request retry. This feature extends write data transfers by one DDR clock cycle to transmit a CRC value for the data transferred in the first four cycles of a write burst. DDR4 devices can be configured to check the incoming CRC against the CRC that they calculate and to report an alert condition to the controller on a mismatch. The controller can then retry the failed (and all subsequent) operation(s).

3.8.3 DRAM Scrubbing

The MCU includes two scrub engines that can be configured to access every memory location in a given range and, if a Correctable Error (CE) is detected, write back the corrected contents. The DDR driver configures the scrub engines at initialization, and then run-time software periodically triggers one or more of the engines to perform its preprogrammed scrub operation. The controller manages the scrub operation's priority to minimize its impact on regular traffic.

3.9 Interrupts

The MCU sends four interrupt output signals to the GIC under these conditions: fault, error recovery, general, and Performance Monitoring Unit (PMU) overflow interrupt.

- Fault conditions include CEs and UEs detected in DRAM or internal RAM.
- Error recovery conditions include: UEs and previously poisoned data that cannot be deferred.
- General interrupt conditions include:
  - DRAM temperature events detected by the DRAM itself or by the MCU.
  - Architectural state changes in the MCU.
  - Memory or register access permission check failures.
  - Link error detection and retry.
  - PHY training requests.
  - Scrub engine completion/status.
- A PMU overflow interrupt condition is signaled if any PMU counter wraps to zero, so that the PMU software can effectively implement higher-order counter bits.

Each condition can be separately enabled in the controller. Interrupt condition details are captured in registers that can be queried by interrupt-handler software to collect additional information about the condition.

3.10 Initialization

The MCU is configured and initialized by a combination of system software and DDR driver code. Numerous registers must be initialized; most are based on the number and type of memory devices attached to the DDR channel and the clock frequency at which the channel is operated. The DDR driver code also performs various training operations before enabling normal memory traffic to the MCU.

3.11 Run-Time Operations

After the MCU is initialized and trained, it can begin servicing memory requests. Several software-controlled actions can be performed during normal operation. These include:

- DDR refresh rate adjustments.
- Periodic PHY termination resistor calibration.
- Periodic memory scrubbing.

3.11.1 DDR Refresh Rate Adjustments

The MCU and attached DRAM devices can be reconfigured during normal operation to change the effective refresh rate in reaction to temperature changes detected by the DRAM devices themselves, or through periodic temperature polling of DRAM by the controller.

3.11.2 Periodic Physical Layer (PHY) Termination Resistor Calibration

The MCU performs PHY IO pad recalibration as needed. This operation temporarily interrupts functional traffic flow to adjust termination controls based on the current voltage and temperature conditions.

SoC firmware provides control parameters for this function. For details, contact Ampere Computing.

3.11.3 Periodic Memory Scrubbing

The MCU controller provides two scrub engines that can be configured to scrub various memory regions. System software can trigger one or both scrub engines periodically based on the frequency of visiting every memory location in DRAM.

Altra Max firmware provides control parameters for this function. For details, contact Ampere Computing.

3.12 Reliability, Availability, and Serviceability (RAS)

See *Section 8.5.3, "Memory Controller Unit (MCU) Error Records and Link Error Registers,"* for detailed information about RAS error records associated with the MCUs.

System Block

4

This chapter describes the system block, which contains the low-speed peripherals, On-Chip Memory (OCM), the System Management processor (SMpro), and the Power Management Processor (PMpro). The chapter covers these topics:

4.1 Overview

The system block, shown in *Figure 1-1* in *Chapter 1, "Overview,"* contains these major functional blocks:

- Low-speed interfaces.
- OCM.
- SMpro.
- PMpro.

Applications can directly access most low-speed interfaces. Because applications cannot directly access OCM, SMpro, and PMpro, these functional blocks are not described in detail.

The system block is in the SoC power domain.

4.2 Low-Speed Interfaces

Altra Max processors support these low-speed interfaces:

- I^2C: Nine I^2C ports for system software, and two additional I^2C ports for SMpro and PMpro firmware.
- QSPI: Two QSPI primary ports operate at up to 33 MHz for SPI flash and TPM connectivity and are configurable as secure world or normal world.
- UART: Five UART ports: one is secure and four are configurable as secure world or normal world.
- GPIO: Three sets of eight GPIOs with interrupt capability. Each set (GPIO0-7, GPIO8-15, and GPIO16-23) can be configured for the secure world or the normal world.
- GPI: Eight GPIs are in the normal world and have no interrupt capability. The signal state is readable by software through an RO register.

4.3 Counter and Timers

An Altra Max processor provides one SC, along with these timers:

- Generic timers: Two generic timers comply with the Arm specification. There are four timer frames and one control base. For more information about the timer frames and control base, see *Section 4.3.7, "Timer Frames and Control Base."*
- Watchdog timers: Two watchdog timers, one in the normal world and the other in the secure world.

4.3.1 Address Map

For the address map of the low-speed peripherals, see *Chapter 6, "System Address Spaces."*

4.3.2 Inter-Integrated Circuit (I^2C)

The Advanced High-Performance Bus Controller (AHBC) integrates nine I^2C controllers, running at up to 1 MHz, that support SMBus. These controllers reside on the internal Advanced Peripheral Bus (APB). An additional I^2C controller supports SMpro and an additional I^2C controller supports PMpro.

- The I^2C ports are SMBus 3.0 and PMBus 1.3 capable but without AVSBus support.
- The I^2C ports support multi-primary control. I^2C IOs are powered on a rail, in common with the BMC, to ensure that I^2C remains powered when the BMC is powered.

Each I^2C controller can be configured as either a primary or secondary device. In addition to the I^2C clock and data IO pins, SCL and SDA, each I^2C bus also has an associated SMBus active low ALERT_N IO pin. When an I^2C controller is configured as a primary device, the associated ALERT_N IO pin should be tri-stated and enabled onto one of the internal SPI type interrupts. When configured as a secondary device, the ALERT_N IO pin can be asserted LOW by software to cause an interrupt to the external primary I^2C device.

The I^2C logic provides these registers:

- A configuration register, CFG_ALERT_OEN, described in *Table 4-1*.
- A status register, ALERT_IN, described in *Table 4-2*.
- An interrupt mask register, ALERT_INTMASK, and an interrupt status register to support the SMBus ALERT IO pins.

Table 4-1: CFG_ALERT_OEN Configuration Register

BIT	TYPE	FIELD	DEFAULT	DESCRIPTION
31:9	RO	RSVD	0	Reserved. RAZ/WI
8	RW	ALERT10_OEN	1	Output Enable control for ALERT10_N IO pin.
7	RW	ALERT9_OEN	1	Output Enable control for ALERT9_N IO pin.
6	RW	ALERT8_OEN	1	Output Enable control for ALERT8_N IO pin.
5	RW	ALERT7_OEN	1	Output Enable control for ALERT7_N IO pin.
4	RW	ALERT6_OEN	1	Output Enable control for ALERT6_N IO pin.
3	RW	ALERT5_OEN	1	Output Enable control for ALERT5_N IO pin.
2	RW	ALERT4_OEN	1	Output Enable control for ALERT4_N IO pin.
1	RW	ALERT3_OEN	1	Output Enable control for ALERT3_N IO pin.
0	RW	ALERT2_OEN	1	Output Enable control for ALERT2_N IO pin. 1 – ALERT2_N IO pin tri-stated 0 – ALERT2_N IO pin driven LOW.

The ALERT_N IO pin cannot be driven HIGH. It can only be enabled to drive LOW or tri-stated. A pull-up resistor on the board is required to pull ALERT_N high.

Table 4-2: ALERT_IN Status Register

BIT	TYPE	FIELD	DEFAULT	DESCRIPTION
31:9	RO	RSVD	0	Reserved. RAZ/WI
8	RO	ALERT10_IN	—	Input state of ALERT10_N IO pin.
7	RO	ALERT9_IN	—	Input state of ALERT9_N IO pin.
6	RO	ALERT8_IN	—	Input state of ALERT8_N IO pin.
5	RO	ALERT7_IN	—	Input state of ALERT7_N IO pin.
4	RO	ALERT6_IN	—	Input state of ALERT6_N IO pin.
3	RO	ALERT5_IN	—	Input state of ALERT5_N IO pin.
2	RO	ALERT4_IN	—	Input state of ALERT4_N IO pin.
1	RO	ALERT3_IN	—	Input state of ALERT3_N IO pin.
0	RO	ALERT2_IN	—	Input state of ALERT2_N IO pin. This bit returns the current state of the ALERT2_N IO pin when read.

Table 4-3: ALERT_INTMASK Interrupt Mask Register

BIT	TYPE	FIELD	DEFAULT	DESCRIPTION
31:9	RO	RSVD	0	Reserved. RAZ/WI
8	RW	ALERT10_MASK	1	Interrupt mask for ALERT10_N IO pin.
7	RW	ALERT9_MASK	1	Interrupt mask for ALERT9_N IO pin.
6	RW	ALERT8_MASK	1	Interrupt mask for ALERT8_N IO pin.
5	RW	ALERT7_MASK	1	Interrupt mask for ALERT7_N IO pin.
4	RW	ALERT6_MASK	1	Interrupt mask for ALERT6_N IO pin.
3	RW	ALERT5_MASK	1	Interrupt mask for ALERT5_N IO pin.
2	RW	ALERT4_MASK	1	Interrupt mask for ALERT4_N IO pin.
1	RW	ALERT3_MASK	1	Interrupt mask for ALERT3_N IO pin.
0	RW	ALERT2_MASK	1	Interrupt mask for ALERT2_N IO pin. 0: ALERT2_N interrupt unmasked. 1: ALERT2_N interrupt masked

An ALERT_N interrupt is generated when:

- The associated ALERT_MASK is cleared to 0.
- The associated ALERT_IN is 0.
- The associated ALERT_OEN is 1.

4.3.3 Quad Serial Peripheral Interface (QSPI)

An Altra Max processor provides two QSPI ports, at up to 50 MHz, for SPI Flash and TPM connectivity.

No memory map mode is supported. There is no intention to run code directly from NOR Flash.

4.3.4 Universal Asynchronous Receiver/Transmitter (UART)

The SoC provides five UART configurations:

- One four-pin UART for the BMC interface (UART0).
- Four two-pin UARTs for the SMpro, PMpro, EL3, and OS/hypervisor consoles.
- UART4_S is a secure world target.

There is no functional IO sharing among the five UARTs. All ports have dedicated IOs.

The AHBC complex integrates the five UARTs on the internal APB. Four UART instances (UART1, UART2, UART3, and UART4) each implement a simple two-wire transmit/receive interface. The UART0 instance supports a four-wire interface with an option to select the control pair as RTS/CTS or DTR/DSR using the UART_MODE_SEL configuration register.

Table 4-4: UART_MODE_SEL Configuration Register

BIT	TYPE	FIELD	DEFAULT	DESCRIPTION
31:1	RO	RSVD	0	Reserved. RAZ/WI
0	RW	CFG_CTS_PAIR	0	This bit selects the UART handshake pair to be used for UART0. 0: DTR/RTS 1: RTS/CTS

4.3.5 General Purpose Input/Outputs (GPIOs)

An Altra Max processor contains three sets of eight GPIOs with interrupt capability. Each set (GPIO0-7, GPIO8-15, and GPIO16-23) can be configured for the secure world or the normal world.

The GPIOs can be configured as:

- Inputs in which the pin value is read through registers.
- Outputs in which the output value and output enable of the pad are controlled through registers.

To mimic an open drain output, software can set the output value to 0 and drive the output enable when required to drive a 0, or tri-state the pad when required to drive a 1 (for this case, a pull up is required on the board).

When configured as an input, the GPIO can be configured to support external interrupts. The polarity is configurable. Interrupts are routed to the GIC, SMpro, and PMpro. The interrupt should be enabled at one of the three destinations.

4.3.6 General Purpose Inputs (GPIs)

An Altra Max processor contains eight GPIs with no interrupt capability in the normal world. Software must read the signal states using an RO register. The eight GPIs cannot be configured as outputs, and cannot route external interrupts to the GIC, SMpro, and PMpro.

4.3.7 Timer Frames and Control Base

Table 4-5 summarizes the timer frames (CNTBase*N*) and control base (CNTCTLBase).

Table 4-5: Timer Frames and Control Bases

BASE ADDRESS	FRAME	DESCRIPTION	TIMER
0x1000 0270 0000	CNTCTLBase	Control Base	Normal and secure
0x1000 0270 0000	CNTBase0	Timer Frame 0	Normal
0x1000 0270 0000	CNTBase1	Timer Frame 1	Normal
0x1000 0270 0000	CNTBase2	Timer Frame 2	Normal
0x1000 0274 0000	CNTBase3	Timer Frame 3	Secure

Note that the CNTCTLBase frame contains a mixture of normal and secure world accessibility at an individual register level. The frames must comply with SBSA Level 4 requirements and the Armv8 specification.

4.3.8 Watchdog Timers

An Altra Max processor provides two watchdog timer pairs, one for the secure world and the other for the normal world.

The WS0 and WS1 signals must be routed as follows:

- WS0_NS (normal WS0) is routed to the GIC (EL2 interrupt).
- WS1_NS (normal WS1) is routed to the GIC but is an EL3 interrupt.
- WS0_S (secure WS0) is also routed to the GIC as an EL3 interrupt.
- WS1_S (secure WS1) is finally routed to SMpro, which triggers the firmware to reset all CPUs.

Table 4-6: Watchdog Timers

BASE ADDRESS (PRIMARY SOCKET)	DESCRIPTION	NORMAL/SECURE WORLD
0x1000 027C 0000	Watchdog Control Frame	Normal
0x1000 027D 0000	Refresh Frame	Normal
0x1000 027E 0000	Watchdog Control Frame	Secure
0x1000 027F 0000	Refresh Frame	Secure

Refer to SBSA Level 4 For compliance requirements.

4.4 System Management Processor (SMpro)

SMpro contains an Arm M3 microcontroller that handles overall system management support, including:

- System booting, described in *Chapter 10, "Boot Process."*
- BMC interface, described in the document titled Altra Max Platform Hardware Design Specifications.
- Error handling for SoC firmware.

When enabled in eFuse Trusted Management Module (TMM) mode, SMpro handles secure boot. SMpro manages the SoC clock and reset and power-failure detection.

4.5 Power Management Processor (PMpro)

Like SMpro, PMpro contains an Arm M3 microcontroller. In PMpro, the microcontroller handles overall power management support as described in *Chapter 11, "Power Management."*

PMpro offloads extended power management functions such as DVFS. PMpro monitors die temperatures and sensor interfaces to perform over-temperature protection and dynamic power estimation.

PCI Express (PCIe) Subsystem

5

This chapter describes the Peripheral Component Interconnect Express (PCIe) implementation in an Altra Max processor. The chapter covers these topics:

5.1 Overview

An Altra Max processor provides 128 PCIe Gen4 lanes for IO expansion. A 2P system provides 192 PCIe Gen4 lanes. In a 1P system, the 128 lanes are provided in eight PCIe x16 RCs. The 128 lanes can be configured as PCIe x16, x8, and x4. The eight RCs are Type A (RcA). The available bifurcation options support up to 32 PCIe controllers.

CCIXFour of the RcAs are dual-mode, each support either up to x16 PCIe or x16 CCIX lanes. Two of the x16 CCIX controllers can be used to connect to another Altra Max processor in a 2P system.

Altra Max processors support PCIe networking cards at data rates up to 200 Gb/s, along with storage expanders and Non-Volatile Memory Express (NVMe) storage devices, making the PCIe implementation well-suited for big data applications.

5.2 Features

Each PCIe controller features:

- Operation at up to 16 GT/s per lane (PCIe Gen4 controller).
- Support for:
  - 64-bit addressing.
  - 512-byte MPS for x16 and x8 controllers and 256-byte MPS for x4.
  - Full line rate performance per port and in the aggregate, based on 256-byte payload size, up to the available memory bandwidth.
  - Auto-negotiation for link speed, reversal, and polarity.
  - Relaxed Ordering (RO) attribute; inbound completions can pass inbound posted writes.
  - Access Control Services (ACS) features to ensure that uncontrolled Peer-to-Peer (P2P) transactions cannot occur.
  - Address Translation Services (ATS), and Page Request Interface (PRI) to help eliminate the need to pin IO pages.
  - Single-Root IO Virtualization (SR-IOV); Alternative Routing ID (ARI) interpretation.
  - Message Signaled Interrupt eXtended (MSI-X) writes encapsulated with DeviceID.
  - Degraded mode on link failure.
  - Hot-plug support using an external onboard component through an I^2C.
  - Link Power Management states, including L0, L1, L2, L3 and D3 (cold/hot).
  - Legacy 32-bit PCIe mode; 32-bit MMIO space in the lower 4 GB of the system address space.
- No support for:
  - P2P transactions.
  - Integrated Non-Transparent Bridging (NTB).
  - Integrated DMA.

5.3 PCI Express (PCIe) Root Complexes (RCs)

A PCIe RC refers to the interface between the Altra Max cores and multiple PCIe components. An RC can contain a host bridge that "bridges" native CPU bus semantics and PCIe semantics. An RC can contain multiple RPs that can connect off-chip devices over external PCIe links. Each RC in a system belongs to a different PCI Segment Group, sometimes called a PCIe domain.

The RCs feature:

- Support for port bifurcation.
- 4 TB system address space mapping.
- Full line rate performance (aggregate of split ports, if any) at the 256-byte (x4) or 512-byte (x16 and x8) MPS.
- Support for Enhanced Configuration Access Mechanism (ECAM). However, PCI-compatible Configuration Access Mechanism (CAM) is not supported. CAM requires IO space, an unsupported feature.
-

The eight 16-lane RCs are Type A (RcA). Each bifurcates from one x16 to four x4s. Four of the RcAs provide a dual-mode x16 PCIe/CCIX controller. An RcA supports these PCIe topologies, one listed topology per RC:

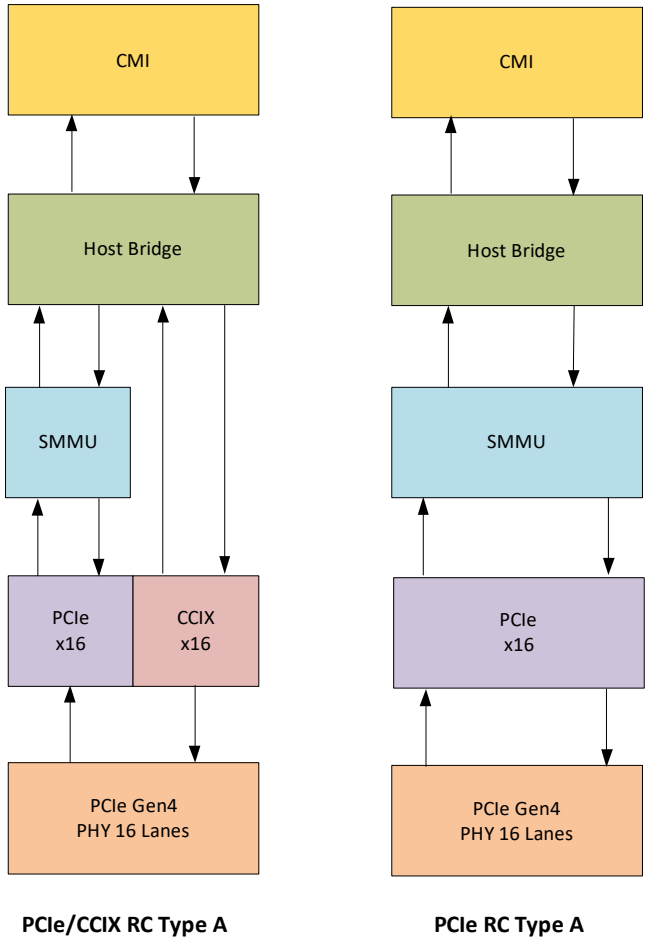
- One x16 PCIe interface.
- Two x8 PCIe interfaces.
- One x8 and two x4 PCIe interfaces.
- Four x4 PCIe interfaces.

For a 2P system, two RcAs per processor each support one x16 Extended Speed Mode (ESM) CCIX controller at transfer rates of up to 25 GT/s. In this use case, no PCIe lanes are available on the x16 CCIX controller.

In a 1P system, the Altra Max processor provides 128 PCIe Gen4 lanes with support for CCIX operation on up to two x16 links. This capability is based on eight 16-lane RCs. A 2P system provides 192 PCIe Gen4 lanes; 64 PCIe lanes are used to interconnect the two processor sockets in the system. 2P connectivity is performed using two RcAs, PCIE0 and PCIE1.

Figure 5-1 provides a block diagram for the PCIe/CCIX and PCIe-only RcAs.

Figure 5-1: PCIe RCs



PCIe/CCIX RC Type A **PCIe RC Type A**

5.4 Host Bridge

The host bridge provides a connection between the CMI and RPs. The host bridge handles accesses to the configuration space (ECAM space, used interchangeably), prefetchable device memory (P-MMIO) and non-prefetchable device memory (NP-MMIO) belonging to each RP hierarchy domain. The host bridge serves as the "root" of the PCIe inverted tree topology, and enables Altra Max cores to communicate with the PCI subsystem.

The host bridge helps to implement an RC that provides multiple RPs. To accomplish this, the host bridge decodes CPU read or write transactions targeted to PCIe and routes them to the appropriate RP. An RC with multiple RPs helps reduce the fragmentation of each region of MMIO space regardless of the number of active controllers in the RC configuration.

The host bridge supports ECAM, 32-bit MMIO, 64-bit MMIO, and Control and Status Register (CSR) accesses.

5.4.1 Host Bridge Reliability, Availability, and Serviceability (RAS)

See *Section 8.5.3.59, "Link Error Interrupt Access Information Register 31:0 (link_err_int_info_31_00),"* for detailed information about RAS error records associated with the PCIe host bridge.

5.5 Input/Output (IO) Virtualization

Altra Max processors support PCIe virtualization with Single-Root IO Virtualization (SR-IOV) devices. Altra Max processors do not support Multi-Root IOV (MR-IOV). Each RC includes one SMMU.

5.5.1 Single Root Input/Output (IO) Virtualization (SR-IOV)

Altra Max processor RPs support SR-IOV capable devices. Some SR-IOV capable EPs also implement ARI to enable the full range of PCIe function numbers. To support these SR-IOV EPs, the PCIe controllers support ARI Forwarding to decode additional function numbers.

5.5.2 StreamID

SMMU v3 uses StreamID and Substream IDs for IO virtualization. A StreamID is connected to the request ID of a PCIe Transaction Layer Packet (TLP). The StreamID is a 16-bit field comprising bus, device, and function numbers.

The SubstreamID is 20-bit field connected to the Process Address Space ID (PASID) field of a PCIe TLP when the system enables the TLP prefix. The PCIe controller forwards the PASID and connects to the SMMU.

5.6 Cache Coherent Interconnect for Accelerators (CCIX) Controllers

Each CCIX controller features:

- A separate Virtual Channel (VC) implemented for CCIX traffic in the x16 controller.
- Support for VDMs and CCIX-optimized TLPs.

To support 2P communication, 128 Request Agent (RA) entries are provided so that all nodes are visible from both sockets.

5.7 Ampere Link Interconnect (ALI)

Altra Max integrates ALI, a proprietary interface for 2P platforms. ALI transparently bypasses the PCIe data link and transaction layers to reduce inter-socket latencies and increase bandwidth. ALI supports PCIe Gen4/CCIX electrical and physical characteristics to maintain platform compatibility with Altra.

ALI provides RAS registers to support enhanced error handling and reporting for 2P platforms. See *Section 8.5.6, "Ampere Link Interconnect (ALI) Error Record Registers,"* for detailed error record register descriptions.

5.8 Hot-Plug Support

Altra Max processors provide native hot-plug support for x4 and x2 PCIe controllers. To support hot-plug for x16 and x8 PCIe controllers, an Altra Max processor must use an external hot-plug controller. The Altra Max processor communicates with the external hot-plug controller using I^2C or GPIO, depending upon platform requirements.

5.9 Interrupts

The PCIe block generates legacy PCI interrupts and four types of Shared Peripheral Interrupts (SPIs):

- INTA/B/C/D: Legacy PCI interrupt.
- UERR: UE interrupt.
- FAULT: Fault (correctable) error interrupt.
- EVENT: Local event, Power Management Event (PME), PCIe defined event.
- ERRINT: Error event, Advanced Error Reporting (AER).

RP interrupts, PCIe-defined and proprietary, are generated as SPI to GIC:

- AER Event: PCIe-defined interrupt.
- PME: Slot Status PME and Root Status PME; PCIe-defined interrupt.
- Firmware Hot Plug Event for PCIe x4 and x2: Hot Plug Event to firmware; subblock level interrupt.
- OS Hot Plug Event for PCIe x4 and x2: Hot Plug Event to OS, PCIe-defined interrupt.
- DTI Error Event: DTI primary device error; controller proprietary interrupt not defined by PCIe.
- RASDP error mode: Controller UE mode, subblock level interrupt.
- Block level event: General block level interrupt, subblock level interrupt.
- Link Autonomous Bandwidth Interrupt: PCIe-defined interrupt.
- Bandwidth Management Interrupt: PCIe-defined interrupt.
- Equalization Request Interrupt: PCIe-defined interrupt.

For more information about Altra Max processor interrupts, see *Chapter 12, "Generic Interrupt Controller (GIC)."*

System Address Spaces

6

This chapter describes the system address spaces of the Altra Max processor. The chapter covers these topics:

6.1 Single-Processor (1P) System Address Space

Table 6-1 describes the 48-bit address space accessible by the CPMs. CSRs are highlighted in yellow and reserved address ranges are highlighted in grey.

Table 6-1: 1P System Address Space (Sheet 1 of 2)

FUNCTION	SIZE	START ADDRESS	END ADDRESS
Reserved	–	0x4000 0000 0000	0x7FFF FFFF FFFF
RcA3 MMCONFIG	256 MB	0x3FFF F000 0000	0x3FFF FFFF FFFF
Reserved	–	0x3FFF E204 0000	0x3FFF EFFF FFFF
RcA3 MSG	256 KB	0x3FFF E200 0000	0x3FFF E203 FFFF
RcA3 CSRs	16 MB	0x3FFF E100 0000	0x3FFF E1FF FFFF
RcA3 TCU	16 MB	0x3FFF E000 0000	0x3FFF E0FF FFFF
RcA3 MMIO	4 TB – 512 MB	0x3C00 0000 0000	0x3FFF DFFF FFFF
RcA2 MMCONFIG	256 MB	0x3BFF F000 0000	0x3BFF FFFF FFFF
Reserved	–	0x3BFF E204 0000	0x3BFF EFFF FFFF
RcA2 MSG	256 KB	0x3BFF E200 0000	0x3BFF E203 FFFF
RcA2 CSRs	16 MB	0x3BFF E100 0000	0x3BFF E1FF FFFF
RcA2 TCU	16 MB	0x3BFF E000 0000	0x3BFF E0FF FFFF
RcA2 MMIO	4 TB – 512 MB	0x3800 0000 0000	0x3BFF DFFF FFFF
RcA1 MMCONFIG	256 MB	0x37FF F000 0000	0x37FF FFFF FFFF
Reserved	–	0x37FF E204 0000	0x37FF EFFF FFFF
RcA1 MSG	256 KB	0x37FF E200 0000	0x37FF E203 FFFF
RcA0 CSRs	16 MB	0x37FF E100 0000	0x37FF E1FF FFFF
RcA1 TCU	16 MB	0x37FF E000 0000	0x37FF E0FF FFFF
RcA1 MMIO	4 TB – 512 MB	0x3400 0000 0000	0x37FF DFFF FFFF
RcA0 MMCONFIG	256 MB	0x33FF F000 0000	0x33FF FFFF FFFF
Reserved	–	0x33FF E204 0000	0x33FF EFFF FFFF
RcA0 MSG	256 KB	0x33FF E200 0000	0x33FF E203 FFFF
RcA0 CSRs	16 MB	0x33FF E100 0000	0x33FF E1FF FFFF
RcA0 TCU	16 MB	0x33FF E000 0000	0x33FF E0FF FFFF
RcA0 MMIO	4 TB – 512 MB	0x3000 0000 0000	0x33FF DFFF FFFF
RcA7 MMCONFIG	256 MB	0x2FFF F000 0000	0x2FFF FFFF FFFF
Reserved	–	0x2FFF E208 0000	0x2FFF EFFF FFFF
RcA7 MSG	512KB	0x2FFF E200 0000	0x2FFF E207 FFFF
RcA7 CSRs	16 MB	0x2FFF E100 0000	0x2FFF E1FF FFFF
RcA7 TCU	16 MB	0x2FFF E000 0000	0x2FFF E0FF FFFF
RcA7 MMIO	4 TB – 512 MB	0x2C00 0000 0000	0x2FFF DFFF FFFF
RcA6 MMCONFIG	256 MB	0x2BFF F000 0000	0x2BFF FFFF FFFF

Table 6-1: 1P System Address Space (Sheet 2 of 2)

FUNCTION	SIZE	START ADDRESS	END ADDRESS
Reserved	–	0x2BFF E2080000	0x2BFF EFFF FFFF
RcA6 MSG	512KB	0x2BFF E200 0000	0x2BFF E207 FFFF
RcA6 CSRs	16 MB	0x2BFF E100 0000	0x2BFF E1FF FFFF
RcA6 TCU	16 MB	0x2BFF E000 0000	0x2BFF E0FF FFFF
RcA6 MMIO	4 TB – 512 MB	0x2800 0000 0000	0x2BFF DFFF FFFF
RcA5 MMCONFIG	256 MB	0x27FF F000 0000	0x27FF FFFF FFFF
Reserved	–	0x27FF E208 0000	0x27FF EFFF FFFF
RcA5 MSG	512KB	0x27FF E200 0000	0x27FF E207 FFFF
RcA5 CSRs	16 MB	0x27FF E100 0000	0x27FF E1FF FFFF
RcA5 TCU	16 MB	0x27FF E000 0000	0x27FF E0FF FFFF
RcA5 MMIO	4 TB – 512 MB	0x2400 0000 0000	0x27FF DFFF FFFF
RcA4 MMCONFIG	256 MB	0x23FF F000 0000	0x23FF FFFF FFFF
Reserved		0x23FF E208 0000	0x23FF EFFF FFFF
RcA4 MSG	512KB	0x23FF E200 0000	0x23FF E207 FFFF
RcA4 CSRs	16 MB	0x23FF E100 0000	0x23FF E1FF FFFF
RcA4 TCU	16 MB	0x23FF E000 0000	0x23FF E0FF FFFF
RcA4 MMIO	4 TB – 512 MB	0x2000 0000 0000	0x23FF DFFF FFFF
Reserved	–	0x1001 0100 0000	0x1FFF FFFF FFFF
GIC	16 MB	0x1001 0000 0000	0x1001 00FF FFFF
SoC IO- SYS	4 GB	0x1000 0000 0000	0x1000 FFFF FFFF
Reserved	–	0x0480 0000 0000	0x0FFF FFFF FFFF
DRAM	8 TB – 2 GB	0x0800 0000 0000 0x0801 0000 0000	0x0800 7FFF FFFF 0x0FFF FFFF FFFF
Reserved	–	0x0001 0000 0000	0x0800 7FFF FFFF
DRAM	2 GB	0x0000 8000 0000	0x0000 FFFF FFFF
RcA3 32-bit MMIO	256 MB	0x0000 7000 0000	0x0000 7FFF FFFF
RcA2 32-bit MMIO	256 MB	0x0000 6000 0000	0x0000 6FFF FFFF
RcA1 32-bit MMIO	256 MB	0x0000 5000 0000	0x0000 5FFF FFFF
RcA0 32-bit MMIO	256 MB	0x0000 4000 0000	0x0000 4FFF FFFF
RcA7 32-bit MMIO	256 MB	0x0000 3000 0000	0x0000 3FFF FFFF
RcA6 32-bit MMIO	256 MB	0x0000 2000 0000	0x0000 2FFF FFFF
RcA5 32-bit MMIO	256 MB	0x0000 1000 0000	0x0000 1FFF FFFF
RcA4 32-bit MMIO	256 MB	0x0000 0000 0000	0x0000 0FFF FFFF

6.2 Dual-Processor (2P) System Address Space

Table 6-2 describes the 48-bit address space accessible by the CPMs in a 2P system. CSRs are highlighted in yellow and reserved address ranges are highlighted in grey.

Table 6-2: 2P System Address Space (Sheet 1 of 4)

SOCKET	FUNCTION	SIZE	START ADDRESS	END ADDRESS
Socket 1	RcA3 (MMIO + TCU + CSRs + MMCNFG)	4 TB	0x7C00 0000 0000	0x7FFF FFFF FFFF
	RcA2 (MMIO + TCU + CSRs + MMCNFG)	4 TB	0x7800 0000 0000	0x7BFF FFFF FFFF
	RcA1 (MMIO + TCU + CSRs + MMCNFG)	4 TB	0x7400 0000 0000	0x77FF FFFF FFFF
	RcA0 (MMIO + TCU + CSRs + MMCNFG)	4 TB	0x7000 0000 0000	0x73FF FFFF FFFF
	RcA7 (MMIO + TCU + CSRs + MMCNFG)	4 TB	0x6C00 0000 0000	0x6FFF FFFF FFFF
	RcA6 (MMIO + TCU + CSRs + MMCNFG)	4 TB	0x6800 0000 0000	0x6BFF FFFF FFFF
	RcA5 (MMIO + TCU + CSRs + MMCNFG)	4 TB	0x6400 0000 0000	0x67FF FFFF FFFF
	RcA4 (MMIO + TCU + CSRs + MMCNFG)	4 TB	0x6000 0000 0000	0x63FF FFFF FFFF
	Reserved	–	0x5001 0400 0000	0x5FFF FFFF FFFF
	GICR + ITS	64 MB–128 KB	0x5001 0002 0000	0x5001 03FF FFFF
	Reserved	128 KB	0x5001 0000 0000	0x5001 0001 FFFF
	SoC IO	4 GB	0x5000 0000 0000	0x5000 FFFF FFFF
	Reserved	–	0x4480 0000 0000	0x4FFF FFFF FFFF
	DRAM	8 TB–1 GB	0x4000 0000 0000 0x4001 0000 0000	0x4000 BFFF FFFF 0x47FF FFFF FFFF
	Reserved	–	0x4000 0000 0000	0x4080 7FFF FFFF
Socket 0	RcA3 MMCONFIG	256 MB	0x3FFF F000 0000	0x3FFF FFFF FFFF
	Reserved	–	0x3FFF E204 0000	0x3FFF EFFF FFFF
	RcA3 MSG	256 KB	0x3FFF E200 0000	0x3FFF E203 FFFF
	RcA3 CSRs	16 MB	0x3FFF E100 0000	0x3FFF E1FF FFFF
	RcA3 TCU	16 MB	0x3FFF E000 0000	0x3FFF E0FF FFFF
	RcA3 MMIO	4 TB–512 MB	0x3C00 0000 0000	0x3FFF DFFF FFFF
	RcA2 MMCONFIG	256 MB	0x3BFF F000 0000	0x3BFF FFFF FFFF
	Reserved	–	0x3BFF E204 0000	0x3BFF EFFF FFFF
	RcA2 MSG	256 KB	0x3BFF E200 0000	0x3BFF E203 FFFF
	RcA2 CSRs	16 MB	0x3BFF E100 0000	0x3BFF E1FF FFFF
	RcA2 TCU	16 MB	0x3BFF E000 0000	0x3BFF E0FF FFFF
	RcA2 MMIO	4 TB-512 MB	0x3800 0000 0000	0x3BFF DFFF FFFF

Table 6-2: 2P System Address Space (Sheet 2 of 4)

SOCKET	FUNCTION	SIZE	START ADDRESS	END ADDRESS
Socket 0	RcA1 MMCONFIG	256 MB	0x37FF F000 0000	0x37FF FFFF FFFF
	Reserved		0x37FF E204 0000	0x37FF EFFF FFFF
	RcA1 MSG	256 KB	0x37FF E200 0000	0x37FF E203 FFFF
	RcA0 CSRs	16 MB	0x37FF E100 0000	0x37FF E1FF FFFF
	RcA1 TCU	16 MB	0x37FF E000 0000	0x37FF E0FF FFFF
	RcA1 MMIO	4 TB-512 MB	0x3400 0000 0000	0x37FF DFFF FFFF
	RcA0 MMCONFIG	256 MB	0x33FF F000 0000	0x33FF FFFF FFFF
	Reserved		0x33FF E204 0000	0x33FF EFFF FFFF
	RcA0 MSG	256 KB	0x33FF E200 0000	0x33FF E203 FFFF
	RcA0 CSRs	16 MB	0x33FF E100 0000	0x33FF E1FF FFFF
	RcA0 TCU	16 MB	0x33FF E000 0000	0x33FF E0FF FFFF
	RcA0 MMIO	4 TB-512 MB	0x3000 0000 0000	0x33FF DFFF FFFF
	RcA7 MMCONFIG	256 MB	0x2FFF F000 0000	0x2FFF FFFF FFFF
	Reserved		0x2FFF E208 0000	0x2FFF EFFF FFFF
	RcA7 MSG	512 KB	0x2FFF E200 0000	0x2FFF E207 FFFF
	RcA7 CSRs	16 MB	0x2FFF E100 0000	0x2FFF E1FF FFFF
	RcA7 TCU	16 MB	0x2FFF E000 0000	0x2FFF E0FF FFFF
	RcA7 MMIO	4 TB-512 MB	0x2C00 0000 0000	0x2FFF DFFF FFFF
	RcA6 MMCONFIG	256 MB	0x2BFF F000 0000	0x2BFF FFFF FFFF
	Reserved		0x2BFF E2080000	0x2BFF EFFF FFFF
	RcA6 MSG	512 KB	0x2BFF E200 0000	0x2BFF E207 FFFF
	RcA6 CSRs	16 MB	0x2BFF E100 0000	0x2BFF E1FF FFFF
	RcA6 TCU	16 MB	0x2BFF E000 0000	0x2BFF E0FF FFFF
	RcA6 MMIO	4 TB-512 MB	0x2800 0000 0000	0x2BFF DFFF FFFF

Table 6-2: 2P System Address Space (Sheet 3 of 4)

SOCKET	FUNCTION	SIZE	START ADDRESS	END ADDRESS
Socket 0	RcA5 MMCONFIG	256 MB	0x27FF F000 0000	0x27FF FFFF FFFF
	Reserved		0x27FF E208 0000	0x27FF EFFF FFFF
	RcA5 MSG	512 KB	0x27FF E200 0000	0x27FF E207 FFFF
	RcA5 CSRs	16 MB	0x27FF E100 0000	0x27FF E1FF FFFF
	RcA5 TCU	16 MB	0x27FF E000 0000	0x27FF E0FF FFFF
	RcA5 MMIO	4 TB-512 MB	0x2400 0000 0000	0x27FF DFFF FFFF
	RcA4 MMCONFIG	256 MB	0x23FF F000 0000	0x23FF FFFF FFFF
	Reserved		0x23FF E208 0000	0x23FF EFFF FFFF
	RcA4 MSG	512 KB	0x23FF E200 0000	0x23FF E207 FFFF
	RcA4 CSRs	16 MB	0x23FF E100 0000	0x23FF E1FF FFFF
	RcA4 TCU	16 MB	0x23FF E000 0000	0x23FF E0FF FFFF
	RcA4 MMIO	4 TB-512 MB	0x2000 0000 0000	0x23FF DFFF FFFF
	Reserved		0x1001 0100 0000	0x1FFF FFFF FFFF
	GICR + ITS	16 MB-128 KB	0x1001 0002 0000	0x1001 00FF FFFF
Socket 0 and Socket 1	GICD	128 KB	0x1001 0000 0000	0x1001 0001 FFFF
Socket 0	SoC IO- SYS	4GB	0x1000 0000 0000	0x1000 FFFF FFFF
	Reserved		0x0480 0000 0000	0x0FFF FFFF FFFF
	DRAM	8 TB	0x0800 0000 0000 0x0800 C000 0000	0x0800 7FFF FFFF 0x0FFF FFFF FFFF
	Reserved		0x0080 0000 0000	0x07FF FFFF FFFF
	Reserved		0x0001 0000 0000	0x007F FFFF FFFF
Socket 1	DRAM – Socket 1	1 GB	0x0000 C000 0000	0x0000 FFFF FFFF
Socket 0	DRAM – Socket 0	1 GB	0x0000 8000 0000	0x0000 BFFF FFFF

Table 6-2: 2P System Address Space (Sheet 4 of 4)

SOCKET	FUNCTION	SIZE	START ADDRESS	END ADDRESS
Socket 1	RcA3 32-bit MMIO	128 MB	0x0000 7800 0000	0x0000 7FFF FFFF
	RcA2 32-bit MMIO	128 MB	0x0000 7000 0000	0x0000 77FF FFFF
	RcA1 32-bit MMIO	128 MB	0x0000 6800 0000	0x0000 6FFF FFFF
	RcA0 32-bit MMIO	128 MB	0x0000 6000 0000	0x0000 67FF FFFF
	RcA7 32-bit MMIO	128 MB	0x0000 5800 0000	0x0000 5FFF FFFF
	RcA6 32-bit MMIO	128 MB	0x0000 5000 0000	0x0000 57FF FFFF
	RcA5 32-bit MMIO	128 MB	0x0000 4800 0000	0x0000 4FFF FFFF
	RcA4 32-bit MMIO	128 MB	0x0000 4000 0000	0x0000 47FF FFFF
Socket 0	RcA3 32-bit MMIO	128 MB	0x0000 3800 0000	0x0000 3FFF FFFF
	RcA2 32-bit MMIO	128 MB	0x0000 3000 0000	0x0000 37FF FFFF
	RcA1 32-bit MMIO	128 MB	0x0000 2800 0000	0x0000 2FFF FFFF
	RcA0 32-bit MMIO	128 MB	0x0000 2000 0000	0x0000 27FF FFFF
	RcA7 32-bit MMIO	128 MB	0x0000 1800 0000	0x0000 1FFF FFFF
	RcA6 32-bit MMIO	128 MB	0x0000 1000 0000	0x0000 17FF FFFF
	RcA5 32-bit MMIO	128 MB	0x0000 0800 0000	0x0000 0FFF FFFF
	RcA4 32-bit MMIO	128 MB	0x0000 0000 0000	0x0000 07FF FFFF

6.3 System-on-Chip (SoC) Input/Output (IO) Address Space

Table 6-2 describes the SoC IO address space. CSRs are highlighted in yellow and reserved address ranges are highlighted in grey. Targets in the secure world appear in green type.

Note: GPIO_23 through GPIO_0 can be in either the normal world or the secure world. Also, the CNTCTL frame of the 256 KB Global Timer supports both secure and non-secure transactions.

Table 6-3: SoC IO Address Space (Sheet 1 of 3)

FUNCTION	SUBFUNCTION	SIZE	START ADDRESS	END ADDRESS	ERROR RESPONSE ON UNPOPULATED ADDRESS
	Reserved	–	0x1000 E000 0000	0x1000 FFFF FFFF	Decode Error (DECERR)
System Debug	Debug	256 MB	0x1000 D000 0000	0x1000 DFFF FFFF	
	Reserved	–	0x1000 D007 1000	0x1000 DFFF FFFF	DECERR
	Reserved	–	0x1000 D004 1000	0x1000 D006 FFFF	DECERR
	Reserved	–	0x1000 D000 0000	0x1000 D003 FFFF	DECERR
DAP	DAP	256 MB	0x1000 D003 FFFF	0x1000 CFFF FFFF	
	Reserved	–	0x1000 C002 1000	0x1000 CFFF FFFF	DECERR
	AXI-Access Port (AP)	4 KB	0x1000 C002 0000	0x1000 C002 0FFF	
	Reserved	–	0x1000 C001 2000	0x1000 C001 FFFF	DECERR
	APB-AP	4 KB	0x1000 C000 1000	0x1000 C000 1FFF	
	Reserved	–	0x1000 C000 1000	0x1000 C001 0FFF	DECERR
	Primary ROM Table	4 KB	0x1000 C000 0000	0x1000 C000 0FFF	
CB	Debug APB	512 MB	0x1000 A000 0000	0x1000 BFFF FFFF	
	CB registers	512 MB	0x1000 8000 0000	0x1000 9FFF FFFF	
	Reserved	–	0x1000 2000 0000	0x1000 7FFF FFFF	DECERR
CMI	CMI Configuration	256 MB	0x1000 1000 0000	0x1000 1FFF FFFF	
	Reserved	–	0x1000 1400 0000	0x1000 1FFF FFFF	Slave Error (SLVERR)
	CMI Configuration	64 MB	0x1000 1000 0000	0x1000 13FF FFFF	
SoC CSRs	Reserved	–	0x1000 0F90 0000	0x1000 0FFF FFFF	DECERR
	OCM	64 KB	0x1000 0F03 0000	0x1000 0F03 FFFF	
	AHBC	64 KB	0x1000 0F02 0000	0x1000 0F02 FFFF	
	PMpro	64 KB	0x1000 0F01 0000	0x1000 0F01 FFFF	
	SMpro	64 KB	0x1000 0F00 0000	0x1000 0F00 FFFF	

Table 6-3: SoC IO Address Space (Sheet 2 of 3)

FUNCTION	SUBFUNCTION	SIZE	START ADDRESS	END ADDRESS	ERROR RESPONSE ON UNPOPULATED ADDRESS
Reserved	Reserved	–	0x1000 0300 0000	0x1000 0EFF FFFF	DECERR
Reserved	Reserved	8 MB	0x1000 0280 0000	0x1000 02FF FFFF	DECERR
AHBC	AHBC	2 MB	0x1000 0260 0000	0x1000 027F FFFF	
	WD Timer 1 S	128 KB	0x1000 027E 0000	0x1000 027F FFFF	
	WD Timer 0	128 KB	0x1000 027C 0000	0x1000 027D FFFF	
	GPIO_16-23	64 KB	0x1000 027B 0000	0x1000 027B FFFF	
	Reserved	–	0x1000 027A 0000	0x1000 027A FFFF	SLVERR
	IIC 10	64 KB	0x1000 0279 0000	0x1000 0279 FFFF	
	IIC 9	64 KB	0x1000 0278 0000	0x1000 0278 FFFF	
	IIC 8	64 KB	0x1000 0277 0000	0x1000 0277 FFFF	
	IIC 7	64 KB	0x1000 0276 0000	0x1000 0276 FFFF	
	IIC 6	64 KB	0x1000 0275 0000	0x1000 0275 FFFF	
	Global Timer S	64 KB	0x1000 0274 0000	0x1000 0274 FFFF	
	Global Timer	256 KB	0x1000 0270 0000	0x1000 0273 FFFF	
	GPIO_0-7	64 KB	0x1000 026F 0000	0x1000 026F FFFF	
	GPIO_8-15	64 KB	0x1000 026E 0000	0x1000 026E FFFF	
	GPI_0-7_NS	64 KB	0x1000 026D 0000	0x1000 026D FFFF	
	IIC 5	64 KB	0x1000 026C 0000	0x1000 026C FFFF	
	IIC 4	64 KB	0x1000 026B 0000	0x1000 026B FFFF	
	IIC 3	64 KB	0x1000 026A 0000	0x1000 026A FFFF	
	IIC 2	64 KB	0x1000 0269 0000	0x1000 0269 FFFF	
	SPI 1	64 KB	0x1000 0268 0000	0x1000 0268 FFFF	
	SPI 0	64 KB	0x1000 0267 0000	0x1000 0267 FFFF	
	Reserved	128 KB	0x1000 0265 0000	0x1000 0266 FFFF	SLVERR
	UART 4_S	64 KB	0x1000 0264 0000	0x1000 0264 FFFF	
	UART 3	64 KB	0x1000 0263 0000	0x1000 0263 FFFF	
	UART 2	64 KB	0x1000 0262 0000	0x1000 0262 FFFF	
	UART 1	64 KB	0x1000 0261 0000	0x1000 0261 FFFF	
	UART 0	64 KB	0x1000 0260 0000	0x1000 0260 FFFF	
Reserved	Reserved	6 MB	0x1000 0200 0000	0x1000 025F FFFF	DECERR

Table 6-3: SoC IO Address Space (Sheet 3 of 3)

FUNCTION	SUBFUNCTION	SIZE	START ADDRESS	END ADDRESS	ERROR RESPONSE ON UNPOPULATED ADDRESS
PMpro	Reserved	10 MB	0x1000 0160 0000	0x1000 01FF FFFF	DECERR
	PMpro	2 MB	0x1000 0140 0000	0x1000 015F FFFF	
	Reserved	–	0x1000 0156 0000	0x1000 015F FFFF	SLVERR
	Reserved	–	0x1000 0080 0000	0x1000 013F FFFF	DECERR
Reserved	Reserved	2 MB	0x1000 0060 0000	0x1000 007F FFFF	DECERR
	SMpro	2 MB	0x1000 0040 0000	0x1000 005F FFFF	
Reserved	Reserved	–	0x1000 0008 0000	0x1000 003F FFFF	DECERR
OCM	OCM	512 KB	0x1000 0000 0000	0x1000 0007 FFFF	

6.4 Root Complex A (RcA) Control and Status Register (CSR) and Message (MSG) Offsets

Table 6-4 describes the RcA CSR and MSG offsets. CSRs are highlighted in yellow and reserved address ranges are highlighted in grey.

Table 6-4: RcA CSR and MSG Offsets

FUNCTION	SIZE	OFFSET START	OFFSET END	GROUP
Reserved	–	0xE150 0000	0xE1FF FFFF	RcA CSRs
TSM	64 KB	0xE140 0000	0xE140 FFFF	
SerDes	2 MB	0xE120 0000	0xE13F FFFF	
Reserved	–	0xE105 0000	0xE11F FFFF	
Pciex4 (i1)	64 KB	0xE104 0000	0xE104 FFFF	
Pciex8	64 KB	0xE103 0000	0xE103 FFFF	
Pciex4 (i0)	64 KB	0xE102 0000	0xE102 FFFF	
Pciex16	64 KB	0xE101 0000	0xE101 FFFF	
Host Bridge	64 KB	0xE100 0000	0xE100 FFFF	
Pciex4 (i1) MSG	64 KB	0xE203 0000	0xE203 FFFF	RcA MSG
Pciex8 MSG	64 KB	0xE202 0000	0xE202 FFFF	
Pciex4 (i0) MSG	64 KB	0xE201 0000	0xE201 FFFF	
Pciex16 MSG	64 KB	0xE200 0000	0xE200 FFFF	

Security

7

This chapter describes Altra Max processor security. The chapter covers these topics:

7.1 Overview

Altra Max processors support these security features:

- All features required by SBSA Level 4 Firmware (EL3, secure memory, secure boot).
- Secure world and normal world main memory and MMIO transactions.
- SMpro as the SoC Root of Trust (RoT) for secure transactions.

The CPMs support the Arm privilege hierarchy of exception levels, from EL0 to EL3, where EL3 is the most privileged. EL3 is intended for a monitor that operates at a higher exception level than EL2, where hypervisors and VMs operate; EL1, for secure and guest operating systems; and EL0, for user applications.

7.2 Design Principles

Altra Max processors implement a security design that embodies these principles, resulting in robust and efficient security features for the processor.

- No security by obscurity.
  - Simply hiding what needs to be protected is not effective. It is prudent to assume that attackers have access to all information needed to mount an attack.
- Apply principles of least privilege.
  - If a hardware or software entity does not require access to a system resource, that access should be blocked.
- Mutual distrust.
  - As an extension of the principle of least privilege, implement and apply "mutual distrust" throughout the system. Trust must be earned through appropriate verification, authentication, certification, and so on.
- Provision for the future.
  - Because implementing new security techniques is expensive, the security architecture (hardware, software, boot, and run-time) must be able to evolve well into the future.
- Comply with standards.
  - Do not reinvent the wheel until it is necessary. Use well-vetted standard cryptography algorithms for encryption and certificate signing. Where appropriate, leverage existing well-established security specifications from organizations such as the Trusted Computing Group (TCG), National Institute of Standards and Technology (NIST), and so on.

7.3 Security Domains

The Armv8 architecture describes two security domains:

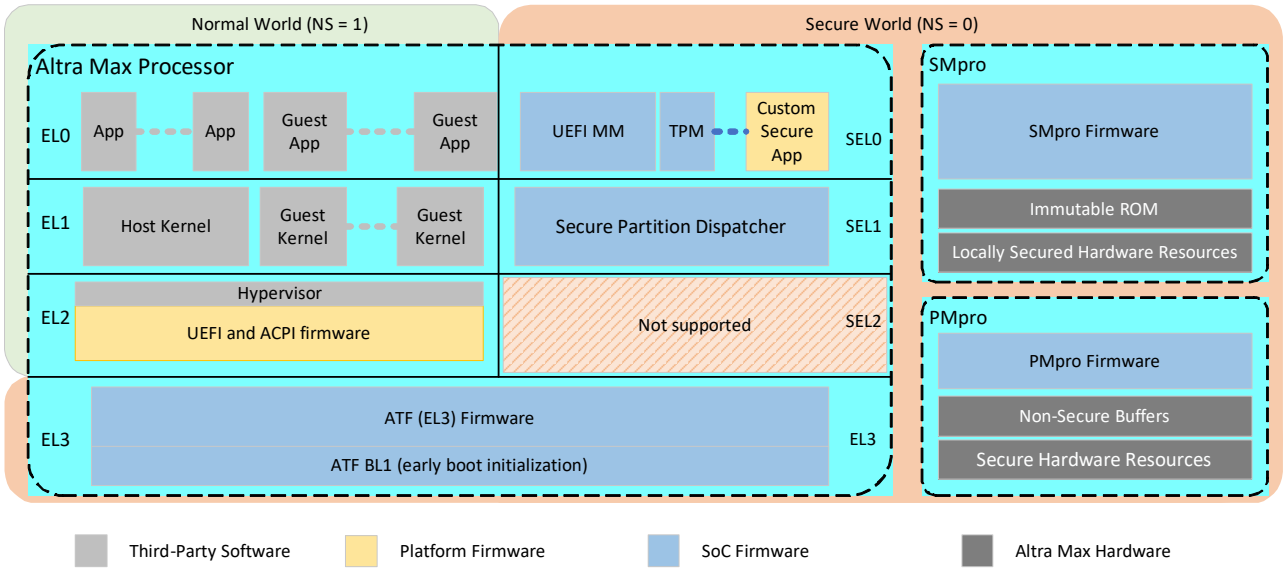
- Normal World – Any firmware that runs in normal world exception levels (EL2, EL1, and EL0).
- Secure World – Any firmware that runs in secure world exception levels (EL3, SEL1, and SEL0).

Note: The architecture also supports a concept of "secure devices and peripherals" that live in the secure world.

Whether an access comes from the secure world or the normal world, a bit is asserted on the bus to indicate which security domain the request comes from. This bit is called the Non-Secure (NS) bit. If NS = 0, the request comes from the secure world. If NS = 1, the request comes from the normal world. Note that requests from the secure world can access resources in the normal world.

Figure 7-1 illustrates the security domains.

Figure 7-1: Security Domains



The normal and secure worlds are represented by an NS bit associated with all accesses. Secure world primary devices can generate accesses with NS=0 and NS=1; normal world primary devices can generate only accesses with NS=1. Architecturally, the NS bit distinguishes between two full 48-bit PA spaces: a 48-bit secure world address space (NS=0) and a 48-bit normal world address space (NS=1). In effect, the NS bit acts as an additional address bit.

7.3.1 Normal World (NS=1)

This is sometimes called the "non-secure world." This is a deprecated term.

These exception levels are in the normal world.

- EL2 typically runs UEFI platform firmware and hypervisor software. If an OS has its hypervisor disabled, at run-time the OS must at least maintain a thin shim layer commonly called a "microvisor."
- EL1 is the "kernel space" in which the main host kernel software typically runs (in Windows, this is called the "root kernel" or "root partition"). This level also runs guest kernel software (in Windows, this is called the "guest partition").
- EL0 is the "user space" in which applications and services software typically runs.

Normal devices are those "owned" by the normal world. Device and configuration registers that are accessible to normal world, and all DMA accesses (if capable), always assert NS=1.

7.3.2 Secure World (NS=0)

This is sometimes called the "TrustZone," a legacy term going back to ARMv6. The Armv8 architecture introduces a more elaborate secure world in which multiple exception levels support secure kernels:

- EL3 is the highest privileged exception level in an Armv8 processor. ATF runs at this level.
- SEL1 typically runs a secure kernel or dispatcher.
- SEL0 typically runs secure applications and services; for example, UEFI MM drivers run in an isolated "stand-alone UEFI MM" app.

7.4 Hardware Security

The Altra Max hardware security architecture relies on three security Life Cycle States (LCSs). Each LCS reflects a TMM eFuse state. At any time, an Altra Max processor is in one of these LCSs. Only the Secure LCS is typically visible to users, and is described in this section. The LCSs not visible to users are Chip Manufacturing (unfused) and Returned Material Authorization (RMA); their descriptions are beyond the scope of this manual.

7.4.1 Secure Life Cycle State (LCS)

Altra Max SoCs that are fully fused when manufactured are in the Secure LCS. Manufacturing eFuses are correctly blown with valid data, including ID fields that provide each Altra Max processor with a unique ID, that is, an HUK. These SoCs are manufactured for large-scale deployments in data centers.

To authenticate firmware images, boot ROM code always runs and leverages either the public key hash in ROM or in TMM eFuses (if revoked). The boot ROM code also programs the One-Time Pad (OTP) CSRs using either a production secret seed from the manufacturing eFuses or from the TMM eFuses (if revoked) for 2P authentication. This is described in more detail in *Section 10.9, "PMpro Boot Responsibilities."*

In the Secure LCS, after a reset, debug is disabled by default.

However, two types of debug certificates can be used in the Secure LCS:

- Debug-disable certificate.
  - This certificate disables Armv8 debug controls (DBGEN, SPIDEN, NIDEN, SPNIDEN). SMpro and PMpro debug are disabled during manufacture when the Altra Max processor is put in the Secure LCS.
  - This certificate can be globally applied (that is, it is not tied to a specific device or firmware build).
  - Multiple debug-disable certificates having different debug masks are supported.
- Debug enable (re-enable) certificate.
  - This certificate is tied to at least one particular firmware image:
    - Re-enabling SMpro/PMpro debug must be tied to the HUK and the SMpro firmware image. If HUK is 0, the SPECIAL_BOOT pin must be asserted.
    - Re-enabling Armv8 secure debug must be tied to the HUK, SMpro image, and ATF BL1 image. If HUK is 0, the SPECIAL_BOOT pin must be asserted.

    - Re-enabling Armv8 non-secure debug can be tied to the HUK and must be tied to a UEFI image. If HUK is 0, the SPECIAL_BOOT pin must be asserted.
  - This certificate provides a debug enable mask.
- Debug certificates re-enabling Armv8 secure debug are delivered to customers upon request.
- Debug certificates re-enabling Armv8 non-secure debug are generated using UEFI build tools provided to customers.
- Debug certificates re-enabling SMpro and PMpro debug are delivered to customers only when Ampere Computing participates in debugging SMpro and PMpro at customer sites.

7.5 Bus Security

Most SoC secondary devices belong wholly either to the secure world or to the normal world. In the former case, the secondary devices are mapped only in the secure address space: they decode the NS bit, process NS=0 accesses normally, and process NS=1 accesses with Read-As-Zero, Writes Ignored (RAZ/WI) behavior. In the latter case, the secondary devices are mapped in both address spaces; they essentially ignore the NS bit for accesses and are equally accessible to both secure world and normal world accesses.

Some SoC secondary devices have more complex responses to secure world or non-secure accesses to each of their addressable locations. The GIC, for example, has some secure world registers, other registers that are in the normal world, and yet other registers that present different views and functionality for secure world and normal world. For these SoC secondary devices, the NS bit is a full-blown attribute that results in accesses to otherwise the same 48b PA to receive different read and/or write behaviors.

The NS bit is carried along with the transactions throughout the complete processor. Primary devices generate the NS bits, and secondary devices then interpret the bits appropriately. Interconnects and bridges do not interpret NS bits but simply pass them along.

Most SoC primary devices are considered part of the normal world and are hardwired to always and only generate normal world accesses. These primary devices, however, can generate secure world transactions:

- Altra Max cores.
- SMpro.
- PMpro.
- DAP.

These secondary devices are always in the secure world, or can be configured to be in the secure world:

- SMpro.
  - D-RAM (not visible if TMM mode is enabled).
  - Secure doorbells.
- Secure UART4.
- GPIO0-7 and/or GPIO8-15 and/or GPIO16-23.
- Secure Global Timer.
- Secure Watchdog Timer.
- PMpro.

  - D-RAM (not visible if TMM mode is enabled).
  - Secure doorbells.
  - I2C0.
- GIC distributor; part of this secondary device is in the secure world and another part is in the normal world. This is handled by hardware.
- Parts of OCM, if enabled.
- TCU; part of this secondary device is in the secure world and another part is in the normal world. This is handled by hardware.
- AHBC peripherals; each peripheral can be independently configured as a secondary device in the secure world or in the normal world using a configuration register in SMpro.

Non-secure accesses to these secure peripherals result in RAZ/WI behavior.

The configuration register address map should be made accessible only to secure world transactions, except for these hard-coded ranges that are accessible in the secure world and in the normal world:

- MCU0: 0xC00 07xx to 0xC00 0Bxx (RAS and PMU).
- MCU1: 0xC40 07xx to 0xC40 0Bxx (RAS and PMU).
- MCU2: 0xC80 07xx to 0xC80 0Bxx (RAS and PMU).
- MCU3: 0xCC0 07xx to 0xCC0 0Bxx (RAS and PMU).
- MCU4: 0xD00 07xx to 0xD00 0Bxx (RAS and PMU).
- MCU5: 0xD40 07xx to 0xD40 0Bxx (RAS and PMU).
- MCU6: 0xD80 07xx to 0xD80 0Bxx (RAS and PMU).
- MCU7: 0xDC0 07xx to 0xDC0 0Bxx (RAS and PMU).

When there is a security violation, the result should be RAZ/WI behavior.

All AHBC CSRs are secure.

7.6 Processor Peripheral Security

Table 7-1 lists the Altra Max processor peripherals, except for PCIe, along with their relationship with the TMM boundary, and shows whether a peripheral is only in the secure world address space.

Table 7-1: Altra Max Processor Peripheral Security (Sheet 1 of 3)

PERIPHERAL	LOCATION	ACCESSIBILITY	INSIDE TMM BOUNDARY	SECURE WORLD ONLY ADDRESS SPACE
I2C0	PMpro	All	—	Yes
I2C1	SMpro	SMpro only	Yes	N/A
I2C2	AHBC	All	—	Configurable
I2C3	AHBC	All	—	Configurable

Table 7-1: Altra Max Processor Peripheral Security (Sheet 2 of 3)

PERIPHERAL	LOCATION	ACCESSIBILITY	INSIDE TMM BOUNDARY	SECURE WORLD ONLY ADDRESS SPACE
I2C4	AHBC	All	—	Configurable
I2C5	AHBC	All	—	Configurable
I2C6	AHBC	All	—	Configurable
I2C7	AHBC	All	—	Configurable
I2C8	AHBC	All	—	Configurable
I2C9	AHBC	All	—	Configurable
I2C10	AHBC	All	—	Configurable
QSPI0	AHBC	All	—	Configurable
QSPI1	AHBC	All	—	Configurable
UART0	AHBC	All	—	Configurable
UART1	AHBC	All	—	Configurable
UART2	AHBC	All	—	Configurable
UART3	AHBC	All	—	Configurable
UART4 (secure)	AHBC	All	—	Yes
GPIO_0-7	AHBC	All	—	Configurable
GPIO_8-15	AHBC	All	—	Configurable
GPIO_16-23	AHBC	All	—	Configurable
GPI_0-7	AHBC	All	—	—
HIGHTEMP_N	System top	All	—	—
OVERTEMP_N	System top	All	—	—
GPIO_FAULT	SMpro	SMpro only	Yes	N/A
CLK_MON_OUT	Top	SMpro only	—	N/A
PCP_PWRCTL	PMpro	All	—	Yes
PCP_PWRGD	PMpro	All	—	Yes
TMM hardware	SMpro	SMpro only	Yes	Yes or N/A
SMpro doorbells (secure world)	SMpro	All	—	Yes
SMpro doorbells (normal world)	SMpro	All	—	—
Global Counter	SMpro	All	—	Some parts are in the secure world address space
SMpro D-RAM and I-RAM	SMpro	All	Yes	Yes

Table 7-1: Altra Max Processor Peripheral Security (Sheet 3 of 3)

PERIPHERAL	LOCATION	ACCESSIBILITY	INSIDE TMM BOUNDARY	SECURE WORLD ONLY ADDRESS SPACE
SMpro CSRs	SMpro	All	Yes	Yes
PMpro doorbells (secure world)	PMpro	All	—	Yes
PMpro doorbells (normal world)	PMpro	All	—	—
PMpro D-RAM and I-RAM	PMpro	All	—	Yes
PMpro CSRs	PMpro	All	—	Yes
Global Timers (secure world)	AHBC	All	—	Yes
Global Timers (normal world)	AHBC	All	—	Some parts are in the secure world address space
Watchdog Timer 0	AHBC	All	—	—
Watchdog Timer 1	AHBC	All	—	Yes
OCM	OCM	All	—	Configurable

7.7 Debug Security

Debug security relies upon JTAG interfaces and Armv8-architected debug domains. If Debug is disabled using eFuses, debug domains can be attacked through the debug domain MMIO interfaces. If Debug is not disabled using eFuses, it is possible to use the JTAG debug interfaces to physically attack the system.

See *Chapter 13, "Debug and Trace,"* for more details about debug domains and JTAG interfaces.

Reliability, Availability, and Serviceability (RAS)

8

This chapter describes RAS features and operation of the Altra Max processor. The chapter covers these topics:

8.1 Overview

This chapter describes how Altra Max processors handle hardware and software errors. This capability is called Reliability, Availability, and Serviceability (RAS):

- Reliability ensures correctness of data by detecting, correcting (where possible), and isolating errors.
- Availability ensures processing capabilities by isolating or disabling a malfunctioning component, enabling the system to continue to operate, although with reduced resources.
- Serviceability includes early detection of faulty components and reduced time to repair or replace them.

The BMC captures error information for all errors, whether reported for hardware, firmware, and software. The BMC logs errors in its System Event Log (SEL). Error sources include:

- SMpro
- PMpro
- Other hardware errors
  - Error logging
  - Error prediction
  - Error recovery
- 2P errors
- ATF
- UEFI

OS and application software errors are outside the scope of this document.

8.1.1 Error Terminology

Table 8-1: Error Terminology

TERM	DEFINITION
Error detection	The initial detection of an error condition. For transmission, errors are not detected. In an agent, a parity or ECC error associated with the internal storage structure can result in an error, as can an illegal access (decode or illegal operation error).
Error handling	The process of responding to a detected error. An attempt is always made to report errors during the processing flow at points at which error handling software can be invoked in the affected context.
Error consumption	The process of using or interpreting part of a data block reported, based on an error bit, to have an error. The error is logged and reported, if error reporting is enabled, at the point of error consumption. For error reporting from SoC agents, an associated interrupt is raised.
Error propagation	Carrying error information with a transaction. The goal of error propagation is to report errors in contexts affected by the error. During error propagation, the scope of the corrupted data size can increase. For example, an error over 32 bytes can be propagated as an error over 64 bytes containing the erroneous 32 bytes. Address-corruption errors are not propagated.
Error reporting	Notifying an error handler through an exception or interrupt. An exception may be synchronous or asynchronous to the request. For asynchronous exceptions and interrupts, an error may be reported in context or out of context of the consumers of the error.
Error termination	Removal of an error from the system by a software error handler.
Error logging	The process of collecting and storing information about the error to enable subsequent diagnostics and repair. An error is logged at the error source.
Error Injection	The process of introducing errors to test system robustness and error handling.

8.1.2 Error Types

Table 8-2: Error Types (Sheet 1 of 2)

ERROR TYPE	SUBCATEGORY	DESCRIPTION
Firmware boot-time error	BMC interface available	During boot-time, error conditions occur and a communication interface is available to provide error information to the next entity.
	BMC interface not available	During boot-time, error conditions occur and no communication interface is available to provide error information to the next entity.
Firmware run-time error	BMC interface available	During run-time, error conditions occur and a communication interface is available to provide error information.
	BMC interface not available	During run-time, error conditions occur and no communication interface is available to provide error information.

Table 8-2: Error Types (Sheet 2 of 2)

ERROR TYPE	SUBCATEGORY	DESCRIPTION
Firmware complete hang	None	The communication entity is locked up. In such cases, the watchdog timer comes into play.
Hardware error	None	Various hardware errors.

8.2 SMpro and Baseboard Management Controller (BMC) Error Interfaces

SMpro and the BMC communicate error information using these error interfaces:

- GPIO interface.

  This interface communicates with the BMC when the I^2C interface is unavailable. For example, only the GPIO interface is available to SMpro ROM firmware.

- I^2C interface.

  This interface provides additional error information to the BMC.

- Silent interface.

  This "interface" refers to a situation in which nothing can run on the SMpro interface. As a result, there is no communication with the BMC. In this case, the BMC detects the condition, logs an event, and reboots the system. If this condition occurs often, the BMC, if configured, can issue a firmware recovery operation.

8.3 Hardware Error Handling

Hardware errors are logged and reported. This enables system designers and users to examine hardware errors in the system for further analysis and actions (such as identify, disable, and replace hardware components).

8.3.1 Hardware Error Handling Features

Altra Max processors provide these error handling features:

- Ability to handle early boot errors without UEFI or ATF.
- Ability to control where error handling is performed:

  A boot variable controls whether error handling is first performed by firmware or the OS.

  - In the firmware-first use case:
    - Offload error handling to SMpro to avoid cycle stealing when possible.
    - Provide error information to the BMC, and to the OS through ACPI Platform Error Interface (APEI).
    - Use Generic Hardware Error Source (GHES v2) error sources to avoid contention among error sources.

  - In the OS-first use case:
    - Provide a Linux kernel driver for parsing Arm Error Source Table (AEST) and handling RAS extension-based errors.
    - Collaborate with OS vendors to develop a similar driver for third-party OSs.
    - Optionally, user space handles BMC error propagation using a RAS daemon.
- Ability to inject errors from the OS using APEI Error Injection (EINJ):
  - True hardware error injection tied to debug fuses.
  - Simulated error injection performed in software.

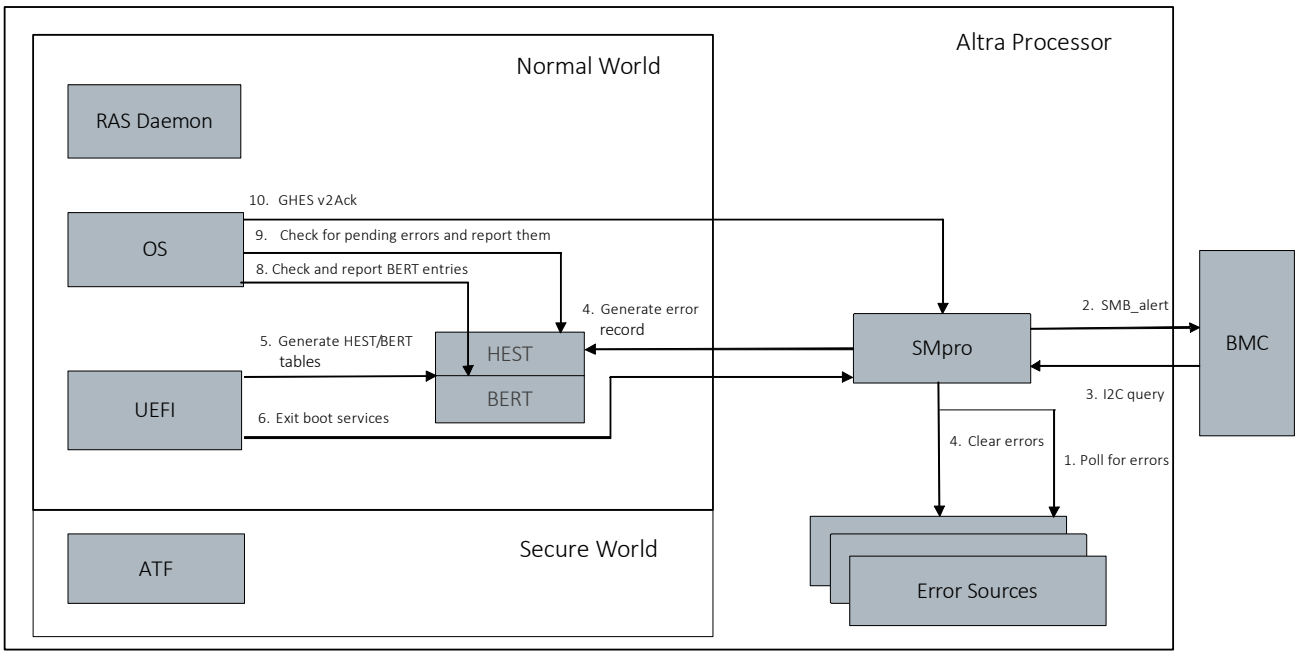
Table 8-3: RAS Hardware Error Handling

	SMPRO	ATF	UEFI MM	OS
Altra Max processor	—	CE/UE	—	CE/UE
Snoop control/SAM	—	CE/UE	—	CE/UE
MCU	CE/UE	—	—	—
PCIe AER	—	—	Secure Partition	CE/UE
PCIe host bridge	CE/UE	—	—	—
GIC	CE/UE	—	—	—
SMMU	CE/UE	—	—	—
OCM	CE/UE	—	—	—
SMpro	CE/UE	—	—	—
PMpro	CE/UE	—	—	—
CMI	CE/UE	—	—	—

8.3.2 Firmware Boot-Time Hardware Error Handling

After SMpro boots, it polls hardware error sources for pending errors. UEFI provides the required ACPI tables, including HEST, AEST, BERT, and EINJ. UEFI populates BERT records pending from the previous boot. If SMpro finds an error during the polling period, SMpro queues the error locally, sends error information to the BMC, and clears the error in hardware. At UEFI exit boot services, SMpro reports locally queued errors to the HEST. When the OS boots, it reports error records in the BERT and reports or handles any error records in the HEST.

Figure 8-1: Firmware Boot-Time Hardware Error Handling

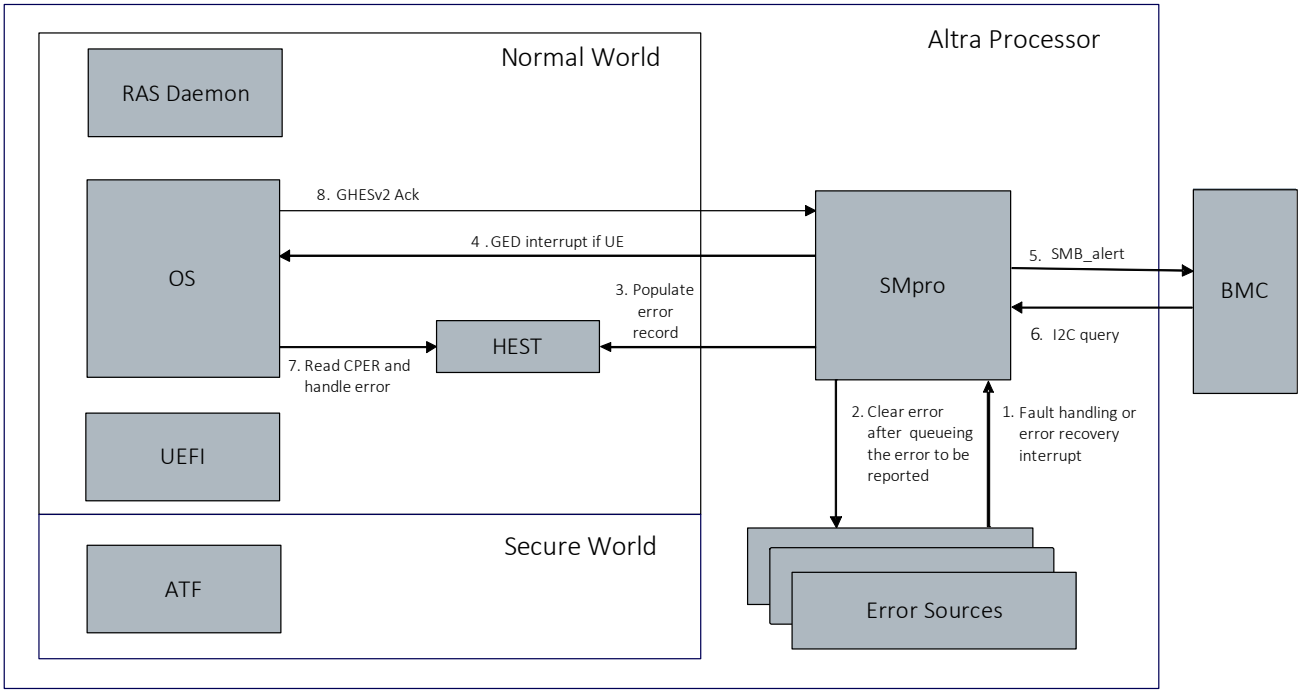


8.3.3 Firmware-First Run-Time Error Handling

When error recovery or fault-handling interrupts for which SMpro is registered trigger, SMpro determines the error sources and queues the error information locally. SMpro updates its I^2C secondary device register map and asserts SMB_ALERT to report the error to the BMC. When the OS is ready (no error acknowledgment is pending for previous errors), SMpro populates the Common Platform Error Records (CPER). If the error is a UE, SMpro triggers a Generic Event Device (GED) interrupt to the OS. The OS polls CEs, so SMpro needs only to populate the CPER. When the OS detects an error, the OS reports the error to the kernel logs, completes any required software intervention (for example, page offline), generates a trace event to user space, and triggers an acknowledgment to SMpro when error reporting finishes.

Figure 8-2 illustrates firmware-first run-time error handling.

Figure 8-2: Firmware-First Run-Time Error Handling



When a CPM error or AER error occurs, the interrupt is sent to ATF. ATF determines the source of the error, populates the CPER record (UEFI MM performs CPER record generation for AER), notifies SMpro of the error so that SMpro can report the error to BMC. ATF then triggers a GED interrupt to the OS. The OS then reports the error to the kernel logs, completes any required software intervention, for example, kills the offending processes or PCIe slot reset, generates a trace event to user space, and triggers an acknowledgment to ATF when error reporting finishes.

Figure 8-3 illustrates firmware-first abort and CPM hardware error handling.

Figure 8-3: Firmware-First Abort and CPM Hardware Error Handling

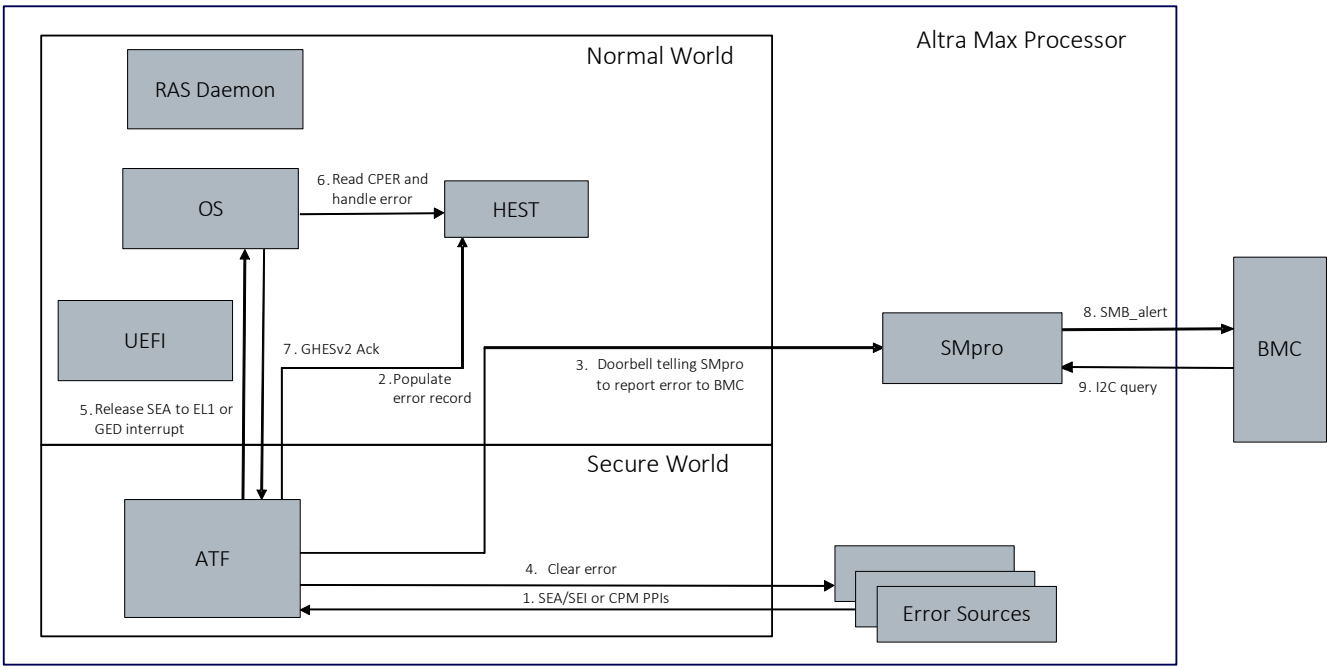
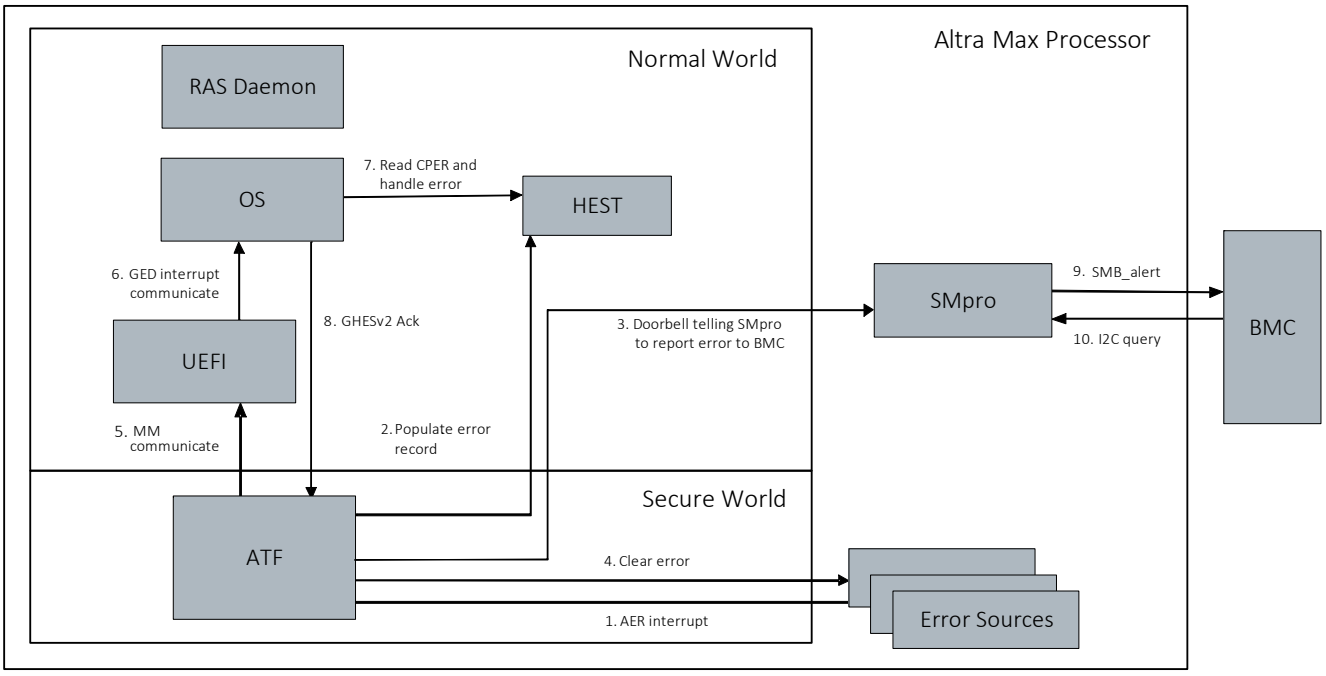


Figure 8-4 illustrates firmware-first abort and hardware AER handling.

Figure 8-4: Firmware-First Abort and Hardware AER Handling



The firmware-first implementation uses GHES v2 error sources defined in the HEST.

Table 8-4: HEST Error Sources

ERROR SOURCE	TYPE	NOTIFICATION TYPE	HARDWARE ERROR SOURCES
0	GHES v2	System Control Interrupt (SCI)	Primary device SMpro UEs
1	GHES v2	Polled	Primary device SMpro CEs
2	GHES v2	SCI	Primary device SMpro CEs
3	GHES v2	SCI	Secondary device SMpro UEs
4	GHES v2	Polled	Secondary device SMpro CEs
5	GHES v2	SCI	Secondary device SMpro CEs
6	GHES v2	SCI	AER-only UEs
7	GHES v2	Polled	AER-only CEs

The error sources define the hardware errors using CPER. Each error is described using the error record section type that is most closely associated with the hardware error source.

Table 8-5: CPER Section Types

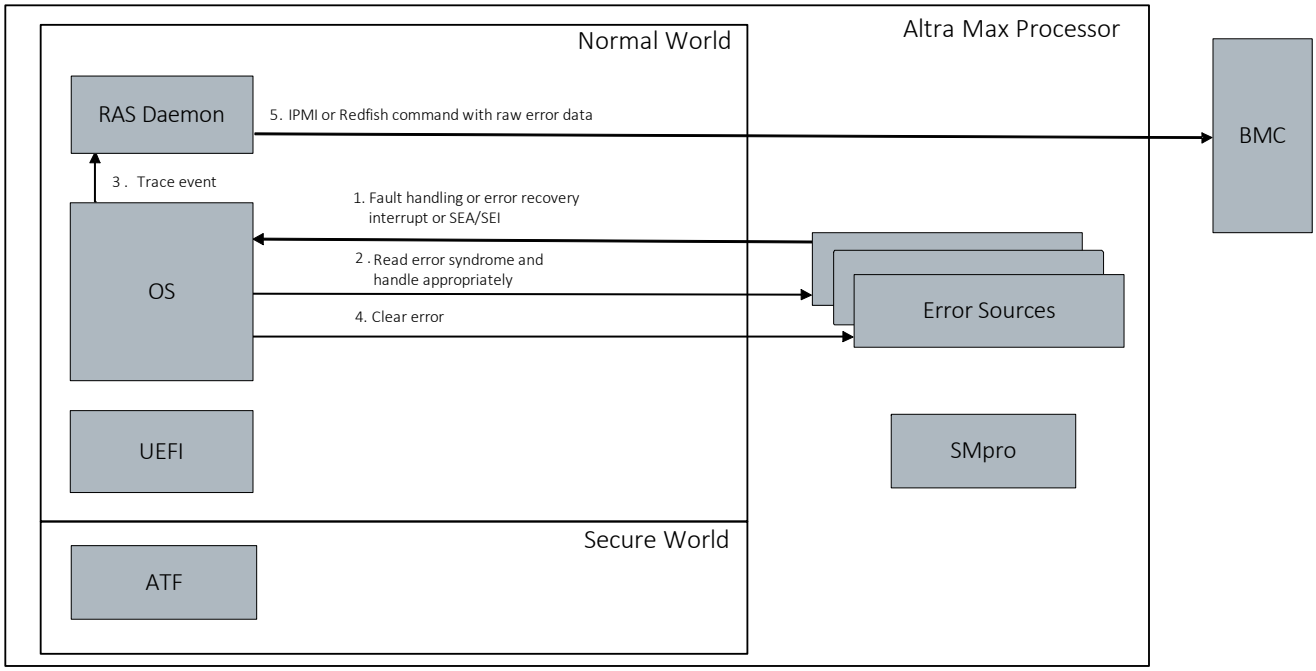
ERROR SOURCE	CPER SECTION TYPE	REFERENCE	NOTES
CPU Cache/Bus/TLB	Altra Max processor	UEFI 2.7 section N.2.4.4	—
DDR	Platform memory	UEFI 2.7 section N.2.5	—
PCIe AER	PCIe	UEFI 2.7 section N.2.7	—
All other sources	Nonstandard	UEFI 2.7 section N.2.3	Raw hexadecimal buffers for each error type must be defined.

8.3.4 Operating System (OS)-First Run-Time Error Handling

When an error recovery, fault handling, or AER interrupt for which the OS is registered triggers, the OS determines the error source and logs the error in the kernel logs. If the error is fatal or uncontainable, the kernel panics. An error is uncontainable if the error can be silently propagated and cannot be isolated to an application or VM, or both. If the error is recoverable or containable with valid address information, the kernel attempts to recover with page offlining, PCIe link resetting, and so on. In non-fatal cases, the OS triggers a trace event to notify user space and clear the error in hardware. Optionally, the RAS daemon propagates errors to the BMC.

Figure 8-5 illustrates OS-first error/abort hardware error handling.

Figure 8-5: OS-First Error/Abort Hardware Error Handling



8.3.5 PCI Express (PCIe) Advanced Error Reporting (AER) Information

PCIe AER handling can use a boot variable, and can be handled either OS-first or firmware-first. OS-first error handling is available through the traditional AER driver. The firmware-first handling uses the Arm framework available in ATF/Standalone MM. Firmware-first handling enables customers to customize PCIe AER handling. UEFI adds dedicated error sources in the HEST table, one for PCIe AER UEs and another for PCIe AER CEs.

8.3.6 Catastrophic Errors

Errors which the firmware determines to be catastrophic to the system, such as watchdog timeout, are stored in SPI-NOR flash to be reported through BERT on the next boot. The firmware forces a system reboot after saving appropriate information in SPI-NOR flash.

8.3.7 ACPI Platform Error Interface (APEI) Error Injection (EINJ)

SMpro supports the standard APEI EINJ framework. SMpro supports true hardware error injection, in which SMpro writes to the EINJ hardware registers, and supports simulated error injection, in which SMpro generates a CPER for the desired error type and reports that to the OS.

The abbreviation "EINJ" refers to the APEI EINJ framework. The term "error injection" is spelled out when referring to the general process or action of injecting errors.

The error types are:

- Processor CE
- Processor UE non-fatal
- Processor UE fatal
- Memory CE
- Memory UE non-fatal
- Memory UE fatal
- PCIe CE
- PCIe UE fatal
- PCIe UE non-fatal

True hardware error injection is configurable using a boot variable, but if the debug fuses are blown, only simulated error injection is possible. *Figure 8-6* illustrates EINJ true hardware error injection.

Note: After Step 3, the error handling flow is the same as firmware-first error handling or OS-first error handling flow, depending upon which error handling is enabled for the injected error type.

Figure 8-6: True Hardware EINJ

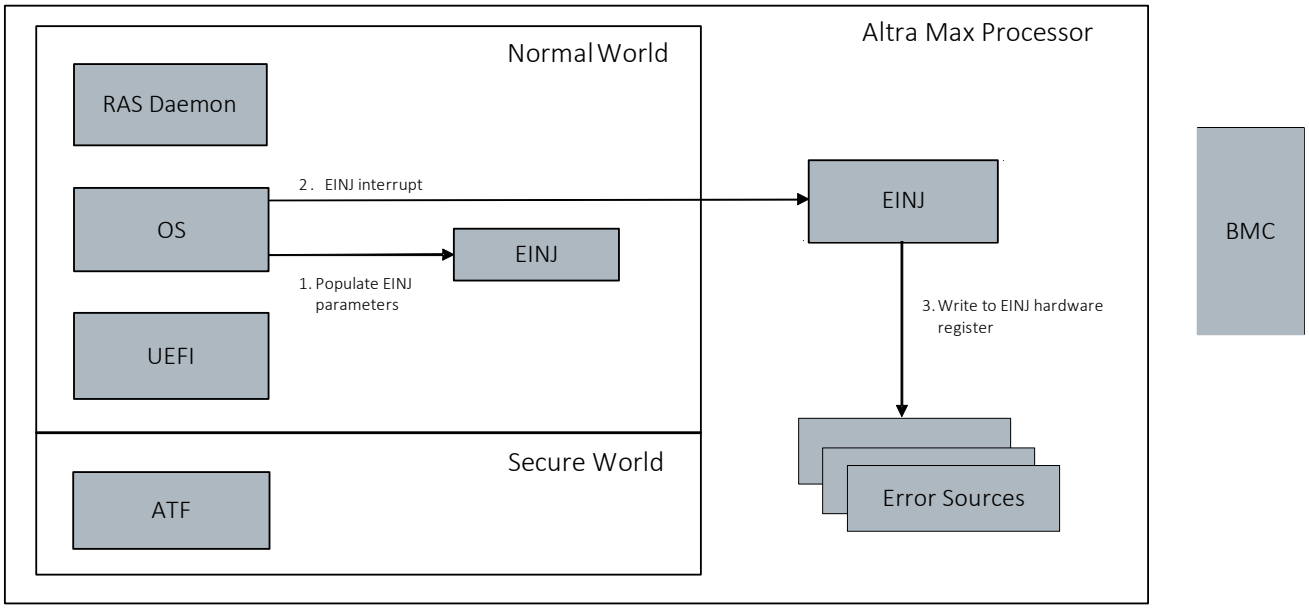


Figure 8-7 illustrates EINJ simulated hardware error injection.

Figure 8-7: Simulated Hardware EINJ

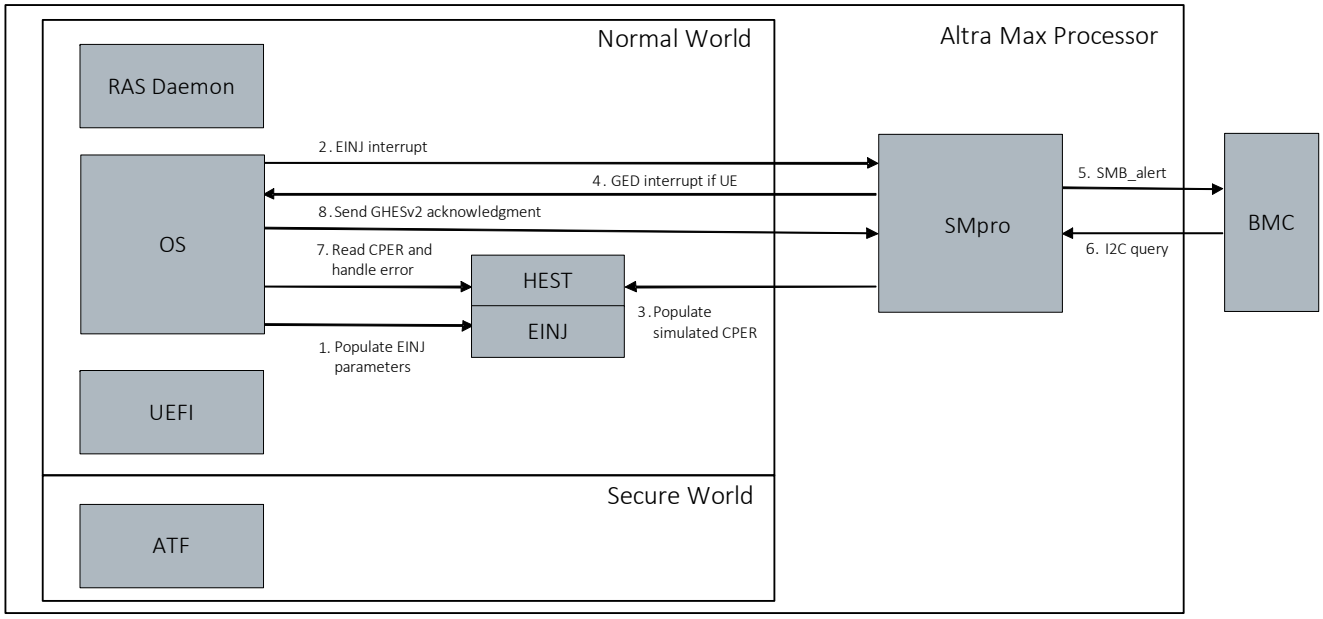


Table 8-6: EINJ Capabilities

ERROR SOURCE	EINJ SOFTWARE-ONLY	EINJ HARDWARE ERROR	STATUS EINJ	ADDRESS EINJ
CPM	Y	Y	Y	—
MCU	Y	Y	Y	Y
PCIe AER	Y	—	Y	Y
OCM	—	—	Y	—
SMpro	—	—	Y	—
PMpro	—	—	Y	—
CMI	—	—	Y	—
GIC	—	—	Y	—
SMMU	—	—	Y	—
Host Bridge/PCIe	—	—	Y	—

8.3.8 Reliability, Availability, and Serviceability (RAS) Configuration

The RAS Configuration menu in the Aptio® Setup Utility UI enables users to modify the default RAS configuration. Configuration options include switching between hardware error injection support and software simulated error injection support, setting CE threshold values, and controlling how certain hardware component errors are handled.

In addition to boot variable configurations, users can customize PCIe AER handling in firmware. The PCIe RAS Secure Partition contains the firmware-first implementation for PCIe AER. Users can edit the partition when specific PCIe adapters require custom error handling techniques.

8.4 Dual-Processor (2P) Error Handling

In 2P systems, errors that occur before 2P sideband I^2C communication is established are handled in the same way as in 1P systems. After the 2P sideband I^2C communication is established, however, secondary device errors are communicated over the 2P sideband I^2C communication channel.

These potential error conditions are specific to 2P systems:

- Both sockets assumed to be primary.
- Secondary device present incorrectly deasserted.
- 2P run-time errors.
- Secondary device errors during boot.
- Primary device errors during boot.

8.4.1 Both Sockets Appear to Be Primary

If both sockets appear to be primary, the system should fail gracefully.

Table 8-7: Potential Conditions Causing Both Sockets to Appear to Be Primary (Sheet 1 of 2)

NUMBER	CONDITIONS	DESCRIPTION
1	Both sockets fail to fetch EEPROM	Socket 0 asserts GPIO_FAULT. Socket 1 asserts GPIO_FAULT.
2	Socket 0 Fetches EEPROM Socket 1 Fails to fetch EEPROM	Socket 0 continues to boot. Socket 1 asserts GPIO_FAULT; Socket 0 detects that Socket 1 Faulted because of a 2P I^2C communication failure.

Table 8-7: Potential Conditions Causing Both Sockets to Appear to Be Primary (Sheet 2 of 2)

NUMBER	CONDITIONS	DESCRIPTION
3	Socket 1 Fetches EEPROM Socket 0 Fails to fetch EEPROM	Socket 0 asserts GPIO_FAULT. Socket 0 Continues to boot. Socket 1 detects that it faulted because of a 2P I^2C communication failure, but continues to load PMpro firmware and ATF B1. Socket falsely assumes that it is the primary. This condition cannot be detected because firmware cannot determine that the 2P GPIO pins can be trusted.
4	Socket 0 and socket 1 Fetch EEPROM	Both sockets continue to boot and fail to detect the secondary socket. In this case, BMC fails to communicate to each socket because the I^2C secondary device uses the same Socket 0 address. The expected BMC behavior is to reset the system and log an event.
5	Both sockets fail to fetch ATF	Exists only with Condition 3; both sockets inform BMC over the BMC I^2C interface. In this case, BMC fails to communicate to each socket because the I^2C secondary device uses the same address. The expected BMC behavior is to reset the system and log an event.
6	Socket 0 Fetches ATF Socket 1 Fails to fetch ATF	Exists only with Condition 3; Socket 1 informs BMC over the BMC I^2C interface. In this case, BMC fails to communicate to each socket because the I^2C secondary device uses the same address. The expected BMC behavior is to reset the system and log an event.
7	Socket 1 Fetches ATF Socket 0 Fails to fetch ATF	Exists only with Condition 3; this does not occur in the Altra Max implementation because the SPI access mux does not switch to Socket 1.
8	Both socket fetches ATF	Exists only with Condition 3; this does not occur in the Altra Max implementation because the SPI access mux does not switch to Socket 1.

8.4.2 Secondary Device Present Incorrect Signal is Deasserted

If the secondary device presents incorrect signal is deasserted, the primary socket assumes that there is no secondary device and the secondary device socket is in reset.

8.4.3 Dual-Processor (2P) Run-Time Errors

The error condition of each socket is communicated over its BMC I^2C interface.

8.4.4 Secondary Device Errors During Boot

The error condition of each socket is communicated over its BMC I^2C interface.

8.4.5 Primary Errors During Boot

The error condition of each socket is communicated over its BMC I^2C interface.

8.4.6 Snoop Control/System Address Map (SAM) Logic Reliability, Availability, and Serviceability (RAS)

The snoop control/SAM logic supports these RAS features:

- Data poisoning on a 64-bit granule.
- Optional ECC protection on RAMs.
- Error recovery and fault handling interrupt outputs.
- Error record registers.

8.5 Error Records

Agents implement one or more error records. When an error is detected, information about the error is written to a record.

An agent might implement multiple error records to record different types of errors, for example, one record for CEs and another record for UEs.

8.5.1 Altra Max Core Error Record Registers

The Altra Max core error record registers are 64-bit unless otherwise noted.

Error records associated with each agent share a set of registers.

Table 8-8: Altra Max Core Error Record Registers (Sheet 1 of 2)

REGISTER	ACCESS	SIZE	DESCRIPTION
ERR0FR	RO	64-bit	Error Record Feature Register
ERR0CTLR	RW	64-bit	Error Record Control Register
ERR0STATUS	RW	32-bit	Error Record Primary Status Register
ERR0ADDR	RW	64-bit	Error Record Address Register

Table 8-8: Altra Max Core Error Record Registers (Sheet 2 of 2)

REGISTER	ACCESS	SIZE	DESCRIPTION
ERR0MISC0	RW	64-bit	Error Record Miscellaneous Register 0
ERR0MISC1	RO	32-bit	Error Record Miscellaneous Register 1

The base simply indicates that there are multiple error records in the system, but the offset and order should be the same for each record.

8.5.1.1 Error Record 0 Feature Register (ERR0FR)

ERR0FR defines which common architecturally defined features are implemented and, of the implemented features, which are software programmable.

ERR0FR is RO.

Table 8-9 summarizes ERR0FR; the Description column provides default field values.

Table 8-9: ERR0FR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
63:18	RO	RSVD	Reserved.
17:16	RO	DUI	Error recovery interrupt for Deferred Error (DE). 00: The core does not support this feature.
15	RO	RP	Repeat counter. 1: A first repeat counter and a second other counter are implemented. The repeat counter is the same size as the primary error counter.
14:12	RO	CEC	CE counter. 010: The node implements an 8-bit standard CE counter in ERR0MISC0[39:32].
11:10	RO	CFI	Fault handling interrupt for CEs. 10: The node implements a control for enabling fault handling interrupts on CEs.
9:8	RO	UE	In-band UE reporting. 01: The node implements in-band UE reporting, that is, external aborts.
7:6	RO	FI	Fault handling interrupt. 10: The node implements a fault handling interrupt and implements controls for enabling and disabling.
5:4	RO	UI	Error recovery interrupt for UEs. 10: The node implements an error recovery interrupt and implements controls for enabling and disabling.

Table 8-9: ERR0FR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
3:2	RO	RSVD	Reserved.
1:0	RO	ED	Error detection and correction. 10: The node implements controls for enabling or disabling error detection and correction.

8.5.1.1.1 Configurations

ERR0FR is accessible from this register when ERRSELR.SEL is 0: ERXFR_EL1.

8.5.1.2 Error Record 0 Control Register (ERR0CTLR)

ERR0CTLR contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling an error recovery interrupt.
- Enabling a fault handling interrupt.
- Enabling error recovery reporting as a read or write error response.

Table 8-10: ERR0CTLR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
63:9	RO	RSVD	Reserved.
8	RW	CFI	Fault handling interrupt for CEs; generated when a standard CE counter on ERR0MISC0 overflows and the overflow bit is set. 0: Fault handling interrupt is not generated for CEs. 1: Fault handling interrupt is generated for CEs. The interrupt is generated, even if the error status is overwritten, because the error record already contains a higher priority error. This applies to reads and writes.
7:4	RO	RSVD	Reserved.
3	RW	FI	Fault handling interrupt enable for detected Deferred Errors (DEs) and UEs. 0: Fault handling interrupt disabled. 1: Fault handling interrupt enabled.
2	RW	UI	Fault handling interrupt enable for detected UEs that are deferred. 0: Fault handling interrupt disabled. 1: Fault handling interrupt enabled.

Table 8-10: ERR0CTLR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
1	RO	RSVD	Reserved.
0	RW	ED	UE recovery interrupt enable for detected UEs that are not deferred. 0: Error recovery interrupt disabled. 1: Error recovery interrupt enabled. This applies to reads and writes.

8.5.1.2.1 Configurations

ERR0CTLR is accessible from this register when ERRSELR.SEL is 0: ERXCTLR_EL1.

8.5.1.3 Error Record 0 Primary Status Register (ERR0STATUS)

ERR0STATUS contains information about the error record:

- Whether any error was detected.
- Whether any detected error was not corrected and returned to a primary device.
- Whether any detected error was not corrected and deferred.
- Whether a second error of the same type was detected before software handled the first error.
- Whether any error was reported.
- Whether other error record registers contain valid information.

Table 8-11: ERR0STATUS Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. 0: ERR0ADDR is not valid. 1: ERR0ADDR contains an address associated with the highest priority error recorded by this record.
30	RW	V	Status Register valid. 0: ERR0STATUS is not valid. 1: ERR0STATUS is valid. At least one error was recorded.
29	RW	UE	UE. 0: No error that cannot be corrected or deferred was detected. 1: At least one error that could not be corrected or deferred was detected. If Error recovery interrupts are enabled, the interrupt signal is asserted until this bit is cleared.
28	RW	ER	Error reported. 0: No external abort was reported. 1: The node reported an external abort to the primary device in access or making a transaction.

Table 8-11: ERR0STATUS Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
27	RW	OF	Overflow. 0: The CE counter did not overflow: • If UE is 1, no error status for a UE was discarded. • If UE is 0 and DE is 1, no error status for a DE was discarded. • If UE is 0, DE is 0, and CE is not 00. 1: More than one error occurred and details of the other error are discarded.
26	RW	MV	Miscellaneous Registers Valid. 0: ERR0MISC0 and ERR0MISC1 are not valid. 1: ERR0MISC0 contains more information about any error recorded in this record.
25:24	RW	CE	CE. 00: No CE was recorded. 10: At least one CE was recorded.
23	RW	DE	DE. 0: No errors were deferred. 1: At least one error was not corrected and deferred by poisoning.
22	RW	PN	Poison. 0: The core cannot distinguish between a poisoned value and a corrupted value.
21:20	RW	UET	UE Type. 00: Uncontainable.
19:5	RO	RSVD	Reserved.
4:0	RW	SERR	Primary error code. 0x0: No error. 0x1: Errors caused by fault injection. 0x2: ECC error from internal data buffer. 0x6: ECC error on cache data RAM. 0x7: ECC error on cache tag or dirty RAM. 0x8: Parity error on TLB data RAM. 0x12: Error response for a cache copy-back. 0x15: DE from the secondary device is not supported at the consumer. For example, poisoned data received from a secondary device by a primary device that cannot defer the error further.

8.5.1.3.1 Configurations

ERR0STATUS is accessible from this register when ERRSELR.SEL is 0: ERXSTATUS_EL1.

8.5.1.4 Error Record 0 Address Register (ERR0ADDR)

ERR0ADDR stores the address that is associated with a recorded error.

Table 8-12: ERR0ADDR Register Format

BIT	TYPE	FIELD	DESCRIPTION
63	RW	NS	Non-secure attribute. 0: The address is in the secure world. 1: The address is in the normal world.
62:48	RO	RSVD	Reserved.
47:0	RW	PADDR	PA of the recorded location.

8.5.1.4.1 Configurations

ERR0STATUS is accessible from this register when ERRSELR.SEL is 0: ERXSTATUS_EL1.

8.5.1.5 Error Record 0 Miscellaneous Register 0 (ERR0MISC0)

ERR0MISC0 is an error syndrome register. It contains CE counters, information to identify where an error is detected, and other state information not available in the corresponding status and address error record registers.

Table 8-13: ERR0MISC0 Register Format (Sheet 1 of 3)

BIT	TYPE	FIELD	DESCRIPTION
63:48	RO	RSVD	Reserved.
47	RW	OFO	Sticky overflow bit, other. 0: Other counter has not overflowed. 1: Other counter has overflowed. The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.
46:40	RW	CECO	CE count, other. Incremented for each CE that does not match the recorded syndrome.
39	RW	OFR	Sticky overflow bit, repeat. 0: Repeat counter did not overflow. 1: Repeat counter overflowed. The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.
38:32	RW	CECR	CE count, repeat. Incremented for the first recorded error, which also records other syndromes, and again for each CE that matches the recorded syndrome.

Table 8-13: ERR0MISC0 Register Format (Sheet 2 of 3)

BIT	TYPE	FIELD	DESCRIPTION
31:28	RW	WAY	The encoding depends upon the unit from which the error being recorded was detected. L1 d-cache: Indicates which tag RAM way or data RAM way detected the error. The upper two bits are unused. L2 TLB: Indicates which RAM has an error. The possible values are 0 (RAM 1) to 9 (RAM 10). L1 i-cache: Indicates which way has the error. The upper two bits are unused.
27:26	RO	RSVD	Reserved.
25	RW	SUBBANK	Encoding depends upon the unit from which the recorded error is detected. L1 i-cache: Indicates which subbank has the error; valid for instruction data cache. For tag errors, this field is zero.
24:23	RW	BANK	Encoding depends upon the unit from which the recorded error is detected. L2 cache: Indicates which L2 bank detects the error. The uppermost bit is unused. L1 i-Cache: Indicates which bank has error; valid for the instruction data cache. For tag errors, this field is zero.
22:19	RW	SUBARRAY	Encoding depends upon the unit from which the recorded error is detected. L2 cache: Indicates which L2 tag way or data doubleword detects the error. The uppermost bit is unused. L1 d-cache Indicates for L1 data RAM which word detects the error; for L1 tag RAMs, indicates which bank has the error (0b0000: bank0, 0b0001: bank1).

Table 8-13: ERR0MISC0 Register Format (Sheet 3 of 3)

BIT	TYPE	FIELD	DESCRIPTION
18:6	RW	INDEX	Encoding depends upon the unit from which the recorded error is detected. L2 cache: Indicates which index detects the error. The upper bits of the index are unused, depending upon cache size. L1 d-cache Indicates which index detects the error. The upper bits of the index are unused, depending upon cache size. L2 TLB index of TLB RAM: The upper four bits are unused. L1 i-cache Indicates which index has the error. The upper bits of the index are unused, depending upon cache size.
5:4	RW	ARRAY	Encoding depends upon the unit from which the recorded error is detected. L2 cache: Indicates which array has the error. 00: L2 tag RAM. 01: L2 data RAM. 10: TQ data RAM. 11: Coherent Hub Interface (CHI) secondary device Error. L1 d-Cache: Indicates which array detects the error. 00: LS0 copy of tag RAM. 01: LS0 Copy of tag RAM. 10: LS tag RAM. L1 i-Cache: Indicates which array detects the error; data array has a higher priority. 0: Tag. 1: Data.
3:0	RW	UNIT	Indicates which unit detects the error. 0001: L1 i-cache. 0100: L1 d-cache. 1000: L2 cache. 0010: L2 TLB.

8.5.1.5.1 Configurations

ERR0MISC0 is accessible from this register when ERRSELR.SEL is 0: ERXMISC0_EL1.

8.5.1.6 Error Record 0 Miscellaneous Register 1 (ERR0MISC1)

ERR0MISC1 is unused in the Altra Max core and marked as RES0.

8.5.1.6.1 Configurations

ERR0MISC0 is accessible from this register when ERRSELR.SEL is 0: ERXMISC1_EL1.

8.5.2 Snoop Control/System Address Map (SAM) Error Records

Table 8-14: Snoop Control/SAM Error Record Registers

REGISTER	ACCESS	SIZE	DESCRIPTION
ERR1FR	RO	64-bit	Error Record Feature Register
ERR1CTLR	RW	64-bit	Error Record Control Register
ERR1MISC0	RW	64-bit	Error Record Miscellaneous Register 0
ERR1MISC1	RO	64-bit	Error Record Miscellaneous Register 1
ERR1PFGCDNR	RW	32-bit	Error Pseudo Fault Generation Countdown Register
ERR1PFGCTLR	RW	32-bit	Error Pseudo Fault Generation Control Register
ERR1PFGFR	RO	32-bit	Error Pseudo Fault Generation Feature Register
ERR1STATUS	RW	32-bit	Error Record Primary Status Register

8.5.2.1 Error Record 1 Feature Register (ERR1FR)

ERR1FR defines which common architecturally defined features are implemented and, of the implemented features, which are software programmable.

ERR1FR is RO.

Table 8-15 summarizes ERR1FR; the Description column provides default field values.

Table 8-15: ERR1FR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
63:20	RO	RSVD	Reserved.
19:18	RO	CEO	CE Overwrite. 00: Counts CE if a counter is implemented and keeps the previous error status. If the-counter overflows, or no counter is implemented, ERR0STATUS.OF is set to 1.
17:16	RO	DUI	Error recovery interrupt for DEs. 00: The core or cluster does not support this feature.

Table 8-15: ERR1FR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
15	RO	RP	Repeat counter. 1: A first repeat counter and a second other counter are implemented. The repeat counter is the same size as the primary error counter.
14:12	RO	CEC	CE Counter. 010: The node implements an 8-bit standard CE counter in ERR0MISC0[39:32]
11:10	RO	CFI	Fault handling interrupt for CEs. 10: The node implements a control for enabling fault handling interrupts on CEs.
9:8	RO	UE	In-band UE reporting. 01: The node implements in-band UE reporting, that is, external aborts.
7:6	RO	FI	Fault handling interrupt. 10: The node implements a fault handling interrupt and implements controls for enabling and disabling.
5:4	RO	UI	Error recovery interrupt for UEs. 10: The node implements an error recovery interrupt and implements controls for enabling and disabling.
3:2	RO	DE	DE enable. 01: DEs are always enabled.
1:0	RO	ED	Error detection and correction. 10: The node implements controls for enabling or disabling error detection and correction.

8.5.2.1.1 Configurations

ERR1FR is accessible from these registers when ERRSELR.SEL is 1:

- [31:0]: ERXFR.
- [63:32]: ERXFR2.
- ERXFR_EL1.

8.5.2.2 Error Record 1 Control Register (ERR1CTLR)

ERR1CTLR contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling an error recovery interrupt.
- Enabling a fault handling interrupt.
- Enabling error recovery reporting as a read or write error response.

Table 8-16: ERR1CTLR Register Format

BIT	TYPE	FIELD	DESCRIPTION
63:9	RO	RSVD	Reserved.
8	RW	CFI	Fault handling interrupt for CEs; generated when a standard CE counter on ERR1MISC0 overflows and the overflow bit is set. 0: Fault handling interrupt is not generated for CEs. 1: Fault handling interrupt is generated for CEs. The interrupt is generated, even if the error status is overwritten, because the error record already contains a higher priority error. This applies to reads and writes.
7:4	RO	RSVD	Reserved.
3	RW	FI	Fault handling interrupt enable for detected DEs and UEs. 0: Fault handling interrupt disabled. 1: Fault handling interrupt enabled.
2	RW	UI	Fault handling interrupt enable for detected UEs that are deferred. 0: Fault handling interrupt disabled. 1: Fault handling interrupt enabled.
1	RO	RSVD	Reserved.
0	RW	ED	UE recovery interrupt enable for detected UEs that are not deferred. 0: Error recovery interrupt disabled. 1: Error recovery interrupt enabled. This applies to reads and writes.

8.5.2.2.1 Configurations

ERR1CTLR is accessible from these registers when ERRSELR.SEL is 1:

- [31:0]: ERXCTLR.
- [63:32]: ERXCTLR2.
- ERXCTLR_EL1.

8.5.2.3 Error Record 1 Miscellaneous Register 0 (ERR1MISC0)

The ERR1MISC0 is an error syndrome register. It contains CE counters, information to identify where an error is detected, and other state information not available in the corresponding status and address error record registers.

Table 8-17: ERR1MISC0 Register Format (Sheet 1 of 3)

BIT	TYPE	FIELD	DESCRIPTION
63:48	RO	RSVD	Reserved.
47	RW	OFO	Sticky overflow bit, other. 0: Other counter has not overflowed. 1: Other counter has overflowed. The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.
46:40	RW	CECO	CE count, other. Incremented for each corrected error that does not match the recorded syndrome.
39	RW	OFR	Sticky overflow bit, repeat. 0: Repeat counter did not overflow. 1: Repeat counter overflowed. The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.
38:32	RW	CECR	CE count, repeat. Incremented for the first recorded error, which also records other syndromes, and again for each CE that matches the recorded syndrome.
31:28	RW	WAY	The encoding depends upon the unit from which the error being recorded was detected. L1 d-cache: Indicates which tag RAM way or data RAM way detected the error. The upper two bits are unused. L2 TLB: Indicates which RAM has an error. The possible values are 0 (RAM 1) to 9 (RAM 10). L1 i-cache: Indicates which way has the error. The upper two bits are unused.
27:26	RO	RSVD	Reserved.
25	RW	SUBBANK	Encoding depends upon the unit from which the recorded error is detected. L1 i-cache: Indicates which subbank has the error; valid for instruction data Cache. For tag errors, this field is zero.

Table 8-17: ERR1MISC0 Register Format (Sheet 2 of 3)

BIT	TYPE	FIELD	DESCRIPTION
24:23	RW	BANK	Encoding depends upon the unit from which the recorded error is detected. L2 cache: Indicates which L2 bank detects the error. The uppermost bit is unused. L1 i-cache: Indicates which bank has error; valid for instruction data cache. For tag errors, this field is zero.
22:19	RW	SUBARRAY	Encoding depends upon the unit from which the recorded error is detected. L2 cache: Indicates which L2 tag way or data doubleword detects the error. The uppermost bit is unused. L1 d-cache Indicates for L1 data RAM which word detects the error; for L1 tag RAMs, indicates which bank has the error (0b0000: bank0, 0b0001: bank1).
18:6	RW	INDEX	Encoding depends upon the unit from which the recorded error is detected. L2 cache: Indicates which index detects the error. The upper bits of the index are unused, depending upon cache size. L1 d-cache Indicates which index detects the error. The upper bits of the index are unused, depending upon cache size. L2 TLB Index of TLB RAM: The upper four bits are unused. L1 i-cache Indicates which index has the error. The upper bits of the index are unused, depending upon cache size.

Table 8-17: ERR1MISC0 Register Format (Sheet 3 of 3)

BIT	TYPE	FIELD	DESCRIPTION
5:4	RW	ARRAY	Encoding depends upon the unit from which the recorded error is detected. L2 cache: Indicates which array has the error. The possible values are: 00: L2 tag RAM. 01: L2 data RAM. 10: TQ data RAM. 11: CHI Secondary Device Error. L1 d-Cache: Indicates which array detects the error. 00: LS0 copy of tag RAM. 01: LS0 Copy of tag RAM. 10: LS tag RAM. L1 i-Cache Indicates which array detects the error; data array has a higher priority. 0: Tag. 1: Data.
3:0	RW	UNIT	Indicates which unit detects the error. 0001: L1 i-cache. 0100: L1 d-cache. 1000: L2 cache. 0010: L2 TLB.

8.5.2.3.1 Configurations

ERR1MISC0 is accessible from these registers when ERRSELR.SEL is 0: ERXMISC0_EL1.

8.5.2.4 Error Record 1 Miscellaneous Register 1 (ERR1MISC1)

ERR1MISC1 is unused and marked as RES0.

8.5.2.4.1 Configurations

ERR1MISC1 is accessible from these registers when ERRSELR.SEL is 1:

- [31:0]: ERXMISC2.
- [63:32]: ERXMISC3.
- ERXMISC1_EL1.

8.5.2.5 Error Record 1 Pseudo Fault Generation Countdown Register (ERR1PFGCDNR)

ERR1PFGCDNR generates one of the errors enabled in the corresponding ERR1PFGCTL register.

Table 8-18: ERR1PFGCDNR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RW	CDN	Countdown value. The reset value of the Error Generation Counter is used for the countdown.

8.5.2.5.1 Configurations

There are no configuration options.

ERR1PFGCDNR is accessible from these registers when ERRSELR.SEL is 1:

- ERXPFGCDNR.
- ERXPFGCDNR_EL1.

8.5.2.6 Error Record 1 Pseudo Fault Generation Control Register (ERR1PFGCTLR)

The ERR1PFGCTLR register enables controlled fault generation.

Table 8-19: ERR1PFGCTLR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	CDEN	Countdown enable. This bit controls transfers from the value that is held in the ERR1PFGCDNR into the Error Generation Counter and enables this counter to start counting down. 0: The Error Generation Counter is disabled. 1: The value held in the ERR1PFGCDNR register is transferred into the Error Generation Counter. The Error Generation Counter counts down.
30	RW	R	Restartable bit; when it reaches 0, the Error Generation Counter restarts from the ERR1PFGCDNR value or stops. 0: When it reaches 0, the counter stops. 1: When it reaches 0, the counter reloads the value that is stored in ERR1PFGCDNR and starts counting down again.
29:7	RO	RSVD	Reserved.
6	RW	CE	CE generation enable. 0: No CE is generated. 1: A CE might be generated when the Error Generation Counter is triggered.
5	RW	DE	DE generation enable. 0: No DE is generated. 1: A DE might be generated when the Error Generation Counter is triggered.

Table 8-19: ERR1PFGCTLR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
4:2	RO	RSVD	Reserved.
1	RW	UC	Uncontainable error generation enable. The possible values are: 0: No uncontainable error is generated 1: An uncontainable error might be generated when the Error Generation Counter is triggered.
0	RO	RSVD	Reserved.

8.5.2.6.1 Configurations

There are no configuration notes.

ERR1PFGCTLR is accessible from these registers when ERRSELR.SEL is 1:

- ERXPFGCTLR.
- ERXPFGCTLR_EL1.

8.5.2.7 Error Record 1 Pseudo Fault Generation Feature Register (ERR1PFGFR)

ERR1PFGFR defines which fault generation features are implemented.

Table 8-20: ERR1PFGFR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RO	PFG	Pseudo Fault Generation. The possible values are: 0: The node does not support fault injection. 1: The node implements a fault injection mechanism.
30	RO	R	Restartable bit; when it reaches 0, the Error Generation Counter restarts from the ERR1PFGCDNR value or stops. 0: The node does not support this feature. 1: This feature is controllable.
29:7	RO	RSVD	Reserved.
6	RO	CE	CE generation enable. 0: The node does not support this feature. 1: This feature is controllable.
5	RO	DE	DE generation enable. 0: The node does not support this feature. 1: This feature is controllable.
4	RO	RSVD	Reserved.

Table 8-20: ERR1PFGFR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
3	RO	UEO	Latent or Restartable Error generation. 0: The node does not support this feature. 1: This feature is controllable.
2	RO	UER	Signaled or Recoverable Error generation. 0: The node does not support this feature. 1: This feature is controllable.
1	RW	UC	Uncontainable Error generation. 0: The node does not support this feature. 1: This feature is controllable.
0	RO	RSVD	Reserved.

8.5.2.7.1 Configurations

ERR1PFGCTLR is accessible from these registers when ERRSELR.SEL is 1:

- ERXPFGCTLR.
- ERXPFGCTLR_EL1.

8.5.2.8 Error Record 1 Primary Status Register (ERR1STATUS)

ERR1STATUS contains information about the error record:

- Whether any error was detected.
- Whether any detected error was not corrected and returned to a primary device.
- Whether any detected error was not corrected and deferred.
- Whether a second error of the same type was detected before software handled the first error.
- Whether any error was reported.
- Whether other error record registers contain valid information.

Table 8-21: ERR1STATUS Register Format (Sheet 1 of 3)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. 0: ERR0ADDR is not valid. 1: ERR0ADDR contains an address associated with the highest priority error recorded by this record.
30	RW	V	Status Register valid. The possible values are: 0: ERR0STATUS is not valid. 1: ERR0STATUS is valid. At least one error was recorded.

Table 8-21: ERR1STATUS Register Format (Sheet 2 of 3)

BIT	TYPE	FIELD	DESCRIPTION
29	RW	UE	UE. 0: No error that cannot be corrected or deferred was detected. 1: At least one error that could not be corrected or deferred was detected. If error recovery interrupts are enabled, the interrupt signal is asserted until this bit is cleared.
28	RW	ER	Error reported. 0: No external abort was reported. 1: The node reported an external abort to the primary device in access or making a transaction.
27	RW	OF	Overflow. 0: The CE counter did not overflow: • If UE is 1, no error status for a UE was discarded. • If UE is 0 and DE is 1, no error status for a DE was discarded. • If UE is 0, DE is 0, and CE is not 00. 1: More than one error occurred and details of other errors are discarded.
26	RW	MV	Miscellaneous Registers Valid. 0: ERR0MISC0 and ERR0MISC1 are not valid. 1: ERR0MISC0 contains more information about any error recorded in this record.
25:24	RW	CE	CE. 00: No CE was recorded. 10: At least one CE was recorded.
23	RW	DE	DE. The possible values are: 0: No errors were deferred. 1: At least one error was not corrected and deferred by poisoning.
22	RW	PN	Poison. 0: The core cannot distinguish between a poisoned value and a corrupted value.
21:20	RW	UET	UE Type. 00: Uncontainable.

Table 8-21: ERR1STATUS Register Format (Sheet 3 of 3)

BIT	TYPE	FIELD	DESCRIPTION
19:16	RO	RSVD	Reserved.
15:8	RW	IERR	An implementation-defined error code. 0x0: No error, or error on other RAMs. 0x2 Error on an L3 snoop filter RAM.
7:0	RW	SERR	Primary error code. 0x0: No error. 0x2: ECC error from internal data buffer. 0x6: ECC error on cache data RAM. 0x7: ECC error on cache tag or dirty RAM. 0x12: Bus error.

8.5.2.8.1 Configurations

ERR1STATUS is accessible from these registers when ERRSELR.SEL is 1:

- ERXSTATUS.
- ERXSTATUS_EL1.

8.5.3 Memory Controller Unit (MCU) Error Records and Link Error Registers

8.5.3.1 Memory Controller Unit (MCU) Error Records

More details about these registers appear starting at *Section 8.5.3.3, "Error Record 0 Feature Register (ERR0FR)"*

Table 8-22: MCU Error Record Registers (Sheet 1 of 3)

REGISTER	ACCESS	SIZE	DESCRIPTION
ERR0FR	RO	32-bit	Error Record 0 Feature Register 0
ERR0CTLR0	RW	32-bit	Error Record 0 Control Register 0
ERR0CTLR1	RW	32-bit	Error Record 0 Control Register 1
ERR0STATUS	RO	32-bit	Error Record 0 Primary Status Register
ERR1FR	RO	32-bit	Error Record 1 Feature Register 1
ERR1CTLR1	RO	32-bit	Error Record 0 Control Register 1
ERR1STATUS	RW	32-bit	Error Record 1 Primary Status Register
ERR1ADDR0	RW	32-bit	Error Record 1 Address Register 0
ERR1ADDR1	RW	32-bit	Error Record 1 Address Register 1

Table 8-22: MCU Error Record Registers (Sheet 2 of 3)

REGISTER	ACCESS	SIZE	DESCRIPTION
ERR1MISC0	RW	32-bit	Error Record 1 Miscellaneous Register 0
ERR1MISC1	RW	32-bit	Error Record 1 Miscellaneous Register 1
ERR1MISC2	RW	32-bit	Error Record 1 Miscellaneous Register 2
ERR1MISC3	RW	32-bit	Error Record 1 Miscellaneous Register 3
ERR1MISC4	RW	32-bit	Error Record 1 Miscellaneous Register 4
ERR1MISC5	RW	32-bit	Error Record 1 Miscellaneous Register 5
ERR2FR	RO	32-bit	Error Record 2 Feature Register 1
ERR2CTLR	RO	32-bit	Error Record 2 Control Register 1
ERR2STATUS	RW	32-bit	Error Record 2 Primary Status Register
ERR2ADDR0	RW	32-bit	Error Record 2 Address Register 0
ERR2ADDR1	RW	32-bit	Error Record 2 Address Register 1
ERR2MISC0	RW	32-bit	Error Record 2 Miscellaneous Register 0
ERR2MISC1	RW	32-bit	Error Record 2 Miscellaneous Register 1
ERR2MISC2	RW	32-bit	Error Record 2 Miscellaneous Register 2
ERR2MISC3	RW	32-bit	Error Record 2 Miscellaneous Register 3
ERR2MISC4	RW	32-bit	Error Record 2 Miscellaneous Register 4
ERR2MISC5	RW	32-bit	Error Record 2 Miscellaneous Register 5
ERR3FR	RO	32-bit	Error Record 3 Feature Register 1
ERR3CTLR	RO	32-bit	Error Record 3 Control Register 1
ERR3STATUS	RO	32-bit	Error Record 3 Primary Status Register
ERR3ADDR0	RW	32-bit	Error Record 3 Address Register 0
ERR3ADDR1	RW	32-bit	Error Record 3 Address Register 1
ERR3MISC0	RO	32-bit	Error Record 3 Miscellaneous Register 0
ERR3MISC1	RO	32-bit	Error Record 3 Miscellaneous Register 1
ERR4FR	RO	32-bit	Error Record 4 Feature Register 1
ERR4CTLR	RO	32-bit	Error Record 4 Control Register 1
ERR4STATUS	RO	32-bit	Error Record 4 Primary Status Register
ERR4ADDR0	RW	32-bit	Error Record 4 Address Register 0

Table 8-22: MCU Error Record Registers (Sheet 3 of 3)

REGISTER	ACCESS	SIZE	DESCRIPTION
ERR4ADDR1	RW	32-bit	Error Record 4 Address Register 1
ERR4MISC0	RW	32-bit	Error Record 4 Miscellaneous Register 0
ERR4MISC1	RO	32-bit	Error Record 4 Miscellaneous Register 1
ERR4MISC2	RO	32-bit	Error Record 4 Miscellaneous Register 2
ERR5FR	RO	32-bit	Error Record 5 Feature Register 1
ERR5CTLR	RO	32-bit	Error Record 5 Control Register 1
ERR5STATUS	RO	32-bit	Error Record 5 Primary Status Register
ERR5ADDR0	RW	32-bit	Error Record 5 Address Register 0
ERR5ADDR1	RW	32-bit	Error Record 5 Address Register 1
ERR5MISC0	RW	32-bit	Error Record 5 Miscellaneous Register 0
ERR5MISC1	RO	32-bit	Error Record 5 Miscellaneous Register 1
ERR5MISC2	RO	32-bit	Error Record 5 Miscellaneous Register 2
ERR6FR	RO	32-bit	Error Record 6 Feature Register 1
ERR6CTLR	RO	32-bit	Error Record 6 Control Register 1
ERR6STATUS	RO	32-bit	Error Record 6 Primary Status Register
ERR6ADDR0	RW	32-bit	Error Record 6 Address Register 0
ERR6ADDR1	RW	32-bit	Error Record 6 Address Register 1
ERR6MISC0	RW	32-bit	Error Record 6 Miscellaneous Register 0
ERR6MISC1	RO	32-bit	Error Record 6 Miscellaneous Register 1
ERRGSR	RW	32-bit	Error Record Group Status Register

8.5.3.2 Memory Controller Unit (MCU) Link Error Registers

More details about these registers appear starting at *Section 8.5.3.58, "Link Error Count Register (link_err_count)"*

Table 8-23: MCU Link Error Registers

REGISTER	ACCESS	SIZE	DESCRIPTION
link_err_count	RW	32-bit	Link Error Count Register
link_err_int_info_31_00	RO	32-bit	Link Error Interrupt Access Information Register 31:0
ink_err_int_info_63_32	RO	32-bit	Link Error Interrupt Access Information Register 63:32

8.5.3.3 Error Record 0 Feature Register (ERR0FR)

ERR0FR defines features which are common to all RAS error records in this section. Each field defines which architecturally-defined common features are implemented and, of the implemented features, which are software programmable.

ERR0FR is RO.

Table 8-24 summarizes ERR0FR; the Description column provides default field values.

Table 8-24: ERR0FR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:12	RO	RSVD	Reserved.
11:10	RO	CFI	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	DE	Indicates whether errors on writes are deferred. 10: The feature is controllable.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.3.4 Error Record 0 Control Register 0 (ERR0CTLR0)

ERR0CTLR0 is a global control register for MCU RAS functions. ERR0CTLR0 controls features such as ECC type and enable, error reporting, interrupt enable, error deferment, and so on. The setting of a global control record bit affects all records.

Table 8-25: ERR0CTLR0 Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:9	RO	RSVD	Reserved.
8	RW	CFI	When enabled, this interrupt is set when any correctable ERRnMISC counters overflow. 0: Disable CFI interrupt. 1: Enable CFI interrupt.
7:4	RO	RSVD	Reserved.

Table 8-25: ERR0CTLR0 Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
3	RW	FI	Enables the ECC fault handling interrupt if an ECC fault is recorded. 0: Disable ECC fault handling interrupt. 1: Enable ECC fault handling interrupt.
2	RW	UI	Enables UE recovery interrupt. The MCU signals an error recovery interrupt when an uncorrectable ECC error is recognized. 0: Disable UE recovery interrupt. 1: Enable UE recovery interrupt.
1	RW	DE	When enabled, the MCU defers uncorrectable read errors to the consumer, sending an OK response and setting the TXDAT poison flag on the CHI-B interconnect. If this bit is clear, the MCU defaults to non-deferred behavior when encountering an unrecoverable error. 0: Disable defer on reads. 1: Enable defer on reads.
0	RW	ED	Enable error detection at the node. When disabled, the correct DRAM ECC is written for writes, but error detection is disabled on reads. 0: Disable ECC detection on reads. 1: Enable ECC detection on reads.

8.5.3.5 Error Record 0 Control Register 1 (ERR0CTLR1)

ERR0CTLR1 is a global control register for MCU RAS functions. The setting of a global control record bit affects all records.

Table 8-26: ERR0CTLR1 Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:8	RO	RSVD	Reserved.
7	RW	CDI	Ignore poison or defer indication from the CHI. If the MCU CHI is not configured for poison indication, this bit has no effect. 0: Check CHI poison indication. 1: Ignore CHI poison indication.
6	RW	CPI	Ignore parity bits from CHI. If the CHI is not configured for parity, this bit has no effect. 0: Check CHI parity bits. 1: Ignore CHI parity bits.
5	RW	WDE	Write Defer Enable; defer (poison) on error instead of interrupting. Setting this bit poisons errant data to defer writing errors into DRAM. Deferment should be disabled if ECC is not present or is disabled. 0: Disable write defer. 1: Enable write defer.

Table 8-26: ERR0CTLR1 Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
4	RW	RMW	Enable Read-Modify-Write (RMW) operations. Must be enabled if Write DBI or ECC is enabled. 0: Disable RMW. 1: Enable RMW.
3	RW	RETRY	Enables or disables retrying ECC uncorrectable data operations. DRAM UEs can be automatically retried once to determine error transience. 0: Disable retry. 1: Enable retry.
2	RW	WB	Enables or disables write-back of ECC corrected data. 0: Disable write-back. 1: Enable write-back.
1:0	RW	ED	Selects the type of ECC protection, either SECDED or Symbol 00: None (no DRAM ECC). 01: SECDED ECC. 10: Symbol ECC.

8.5.3.6 Error Record 0 Primary Status Register (ERR0STATUS)

ERR0STATUS is reserved.

Table 8-27: ERR0STATUS Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.7 Error Record 1 Feature Register (ERR1FR)

Reading ERR1FR returns the same value programmed in ERR0FR.

ERR1FR is RO.

Table 8-28 summarizes ERR1FR; the Description column provides default field values.

Table 8-28: ERR1FR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:12	RO	RSVD	Reserved.
11:10	RO	CFI	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.

Table 8-28: ERR1FR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	DE	Indicates whether errors on writes are deferred. 10: The feature is controllable.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.3.8 Error Record 1 Control Register (ERR1CTLR)

ERR1CTLR is reserved. ERR0CTLR controls this register. Reading this register returns all zeros.

Table 8-29: ERR1CTLR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.9 Error Record 1 Primary Status Register (ERR1STATUS)

ERR1STATUS reports error type, status, and contains valid bits for extra syndrome registers.

Table 8-30: ERR1STATUS Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. The ERR1ADDR registers for this record contain a PA associated with the highest priority error recorded by this record.
30	RW	V	This register is valid. At least one error was recorded.
29:28	RO	RSVD	Reserved.
27	RW	OF	This bit is directly tied to the CE counter of this record. This field cannot be directly cleared. To clear this bit you must ensure all counter OF bits for this record are clear. 0: No CE counter overflowed. 1: At least one CE counter overflowed.

Table 8-30: ERR1STATUS Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
26	RW	MV	Miscellaneous Registers Valid. The ERR1MISC registers for this record contain more information for an error recorded by this record.
25:24	RW	CE	CE. Indicates whether CEs were recorded. CE[1] is the corrected fault handling interrupt status bit. CE[1] is set when an error is corrected and the corresponding counter overflows. CE[0] does not interrupt. It is set when the counter advances and indicates new syndrome information was captured. 00: None No CEs were recorded. 01: Transient A non-overflowing error was detected and the syndrome for the error was captured. 10: Reserved 11: Persistent An error counter belonging to this record overflowed, asserting a CFH interrupt if enabled. No further syndrome information is captured until CE[1] is cleared. Check the IERR status bits to determine which counter overflowed.
23:16	RO	RSVD	Reserved.
15:8	RW	IERR	Indicates which rank counter overflowed. Bit 0 is the indicator for rank 0, bit 1 is the indicator for rank 1, and so on. This field is set when an error is corrected and the corresponding counter overflows.
19:5	RO	RSVD	Reserved.
7:0	RW	SERR	Architecturally-defined primary error code; indicates the type of error. 0b0000 0000: None (no error). 0b0000 1100: DRAM; ECC error in SDRAM.

8.5.3.10 Error Record 1 Address Register 0 (ERR1ADDR0)

ERR1ADDR0 contains the physical bank, row, and column. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-31: ERR1ADDR0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:28	RW	BANK	Physical bank.
27:10	RW	ROW	Physical row.
9:0	RW	COL	Physical column.

8.5.3.11 Error Record 1 Address Register 1 (ERR1ADDR1)

ERR1ADDR0 Contains the physical Chip ID (CID) and rank. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-32: ERR1ADDR1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:6	RO	RSVD	Reserved.
5:3	RW	CID	Physical CID.
2:0	RW	RANK	Physical rank.

8.5.3.12 Error Record 1 Miscellaneous Register 0 (ERR1MISC0)

ERR1MISC0 provides the physical rank, row, and column of the last error detected before an interrupt is asserted.

Table 8-33: ERR1MISC0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR1STATUS.MV is set and cleared when ERR1STATUS.MV bit is cleared.
30:28	RW	RANK	Physical rank of the last error detected before an interrupt is asserted.
27:10	RW	ROW	Physical row of the last error detected before an interrupt is asserted.
9:0	RW	COL	Physical column of the last error detected before an interrupt is asserted.

8.5.3.13 Error Record 1 Miscellaneous Register 1 (ERR1MISC1)

ERR1MISC1 provides the bank, logical rank, and failed nibble location of the last error detected before an interrupt is asserted.

Table 8-34: ERR1MISC1 Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR1STATUS.MV is set and cleared when ERR1STATUS.MV bit is cleared.
30:28	RW	CID	Indicates which logical rank on a multi-rank device, such as 3DS Through-Silicon Via (TSV) stacks, has the error.
27:24	RW	LOC_VAL	Used in Extended ECC (Reed-Solomon) mode, which can detect and correct up to four random errors. These are the nibble locations of the corrected errors. The fields indicate in which of 18 possible nibble locations (0..17) each error was found. The fields must be qualified with the LOC_VAL bits because fewer than four nibbles may have failed.

Table 8-34: ERR1MISC1 Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
23:19	RW	NIB_LOC3	Used in Extended ECC (Reed-Solomon) mode, which can detect and correct up to four random errors. These are the nibble locations of the corrected errors. The fields indicate in which of 18 possible nibble locations (0..17) each error was found. The fields must be qualified with the LOC_VAL bits because fewer than four nibbles may have failed.
18:14	RW	NIB_LOC2	Used in Extended ECC (Reed-Solomon) mode which can detect and correct up to four random errors. These are the nibble locations of the corrected errors. The fields indicate in which of 18 possible nibble locations (0..17) each error was found. The fields must be qualified with the LOC_VAL bits because fewer than four nibbles may have failed.
13:9	RW	NIB_LOC1	Used in Extended ECC (Reed-Solomon) mode which can detect and correct up to four random errors. These are the nibble locations of the corrected errors. The fields indicate in which of 18 possible nibble locations (0..17) each error is found. The fields must be qualified with the LOC_VAL bits because fewer than four nibbles may have failed.
8:4	RW	NIB_LOC0	Used in Extended ECC (Reed-Solomon) mode which can detect and correct up to four random errors. These are the nibble locations of the corrected errors. The fields indicate in which of 18 possible nibble locations (0..17) each error was found. The fields must be qualified with the LOC_VAL bits because fewer than four nibbles may have failed.
3:0	RW	BANK	Indicates the bank of the last error detected before an interrupt is asserted.

8.5.3.14 Error Record 1 Miscellaneous Register 2 (ERR1MISC2)

ERR1MISC2 contains DRAM CE counters.

Table 8-35: ERR1MISC2 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	RANK1_OF	Rank 1 error overflow.
30:16	RW	RANK1_CT	Rank 1 error count.
15	RW	RANK0_OF	Rank 0 error overflow.
14:0	RW	RANK1_CT	Rank 0 error count.

8.5.3.15 Error Record 1 Miscellaneous Register 3 (ERR1MISC3)

ERR1MISC3 contains DRAM CE counters.

Table 8-36: ERR1MISC3 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	RANK3_OF	Rank 1 error overflow.
30:16	RW	RANK3_CT	Rank 1 error count.
15	RW	RANK2_OF	Rank 0 error overflow.
14:0	RW	RANK2_CT	Rank 0 error count.

8.5.3.16 Error Record 1 Miscellaneous Register 4 (ERR1MISC4)

ERR1MISC4 contains DRAM CE counters.

Table 8-37: ERR1MISC4 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	RANK5_OF	Rank 1 error overflow.
30:16	RW	RANK5_CT	Rank 1 error count.
15	RW	RANK4_OF	Rank 0 error overflow.
14:0	RW	RANK4_CT	Rank 0 error count.

8.5.3.17 Error Record 1 Miscellaneous Register 5 (ERR1MISC5)

ERR1MISC5 contains DRAM CE counters.

Table 8-38: ERR1MISC5 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	RANK7_OF	Rank 1 error overflow.
30:16	RW	RANK7_CT	Rank 1 error count.
15	RW	RANK6_OF	Rank 0 error overflow.
14:0	RW	RANK6_CT	Rank 0 error count.

8.5.3.18 Error Record 2 Feature Register (ERR2FR)

Reading ERR2FR returns the same value programmed in ERR0FR.

ERR2FR is RO.

Table 8-39 summarizes ERR2FR; the Description column provides default field values.

Table 8-39: ERR2FR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:12	RO	RSVD	Reserved.
11:10	RO	CFI	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	DE	Indicates whether errors on writes are deferred. 10: The feature is controllable.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.3.19 Error Record 2 Control Register (ERR2CTLR)

ERR2CTLR is reserved. ERR0CTLR controls this register. Reading this register returns all zeros.

Table 8-40: ERR2CTLR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.20 Error Record 2 Primary Status Register (ERR2STATUS)

ERR2STATUS reports error type, status, and contains valid bits for extra syndrome registers.

Table 8-41: ERR2STATUS Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. The ERR1ADDR register for this record contains a PA associated with the highest priority error recorded by this record.
30	RW	V	This register is valid. At least one error was recorded.

Table 8-41: ERR2STATUS Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
29:28	RO	RSVD	Reserved.
27	RW	OF	This bit is directly tied to the CE counter(s) of this record. This field cannot be directly cleared. To clear this bit you must ensure all counter OF bits for this record are clear. 0: No CE counter overflowed. 1: At least one CE counter overflowed.
26	RW	MV	Miscellaneous Registers Valid. The ERR2MISC registers for this record contains more information for an error recorded by this record.
25:24	RW	CE	CE. Indicates whether CEs were recorded. CE[1] is the corrected fault handling interrupt status bit. CE[1] is set when an error is corrected and the corresponding counter overflows. CE[0] does not interrupt. It is set when the counter advances and indicates new syndrome information was captured. 00: None No CEs were recorded. 01: Transient A non-overflowing error was detected and the syndrome for the error was captured. 10: Reserved 11: Persistent An error counter belonging to this record overflowed, asserting a CFH interrupt if enabled. No further syndrome information is captured until CE[1] is cleared. Check the IERR status bits to determine which counter overflowed.
23:16	RO	RSVD	Reserved.
15:8	RW	IERR	Indicates which rank counter overflowed. Bit 0 is the indicator for rank 0, bit 1 is the indicator for rank 1, and so on. This field is set when an error is corrected and the corresponding counter overflows.
19:5	RO	RSVD	Reserved.
7:0	RW	SERR	Architecturally-defined primary error code; indicates the type of error. 0b0000 0000: None; no error. 0b0000 1100: DRAM; ECC error in SDRAM.

8.5.3.21 Error Record 2 Address Register 0 (ERR2ADDR0)

ERR2ADDR0 contains the physical bank, row, and column. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-42: ERR2ADDR0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:28	RW	BANK	Physical bank.
27:10	RW	ROW	Physical row.
9:0	RW	COL	Physical column.

8.5.3.22 Error Record 2 Address Register 1 (ERR2ADDR1)

ERR2ADDR0 Contains the physical CID and rank. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-43: ERR2ADDR1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:6	RO	RSVD	Reserved.
5:3	RW	CID	Physical CID.
2:0	RW	RANK	Physical rank.

8.5.3.23 Error Record 2 Miscellaneous Register 0 (ERR2MISC0)

ERR2MISC0 provides the physical rank, row, and column of the first error detected since last cleared.

Table 8-44: ERR2MISC0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR2STATUS.MV is set and cleared when ERR2STATUS.MV bit is cleared.
30:28	RW	RANK	Physical rank of the last error detected before an interrupt is asserted.
27:10	RW	ROW	Physical row of the last error detected before an interrupt is asserted.
9:0	RW	COL	Physical column of the last error detected before an interrupt is asserted.

8.5.3.24 Error Record 2 Miscellaneous Register 1 (ERR2MISC1)

ERR2MISC1 provides the bank and logical rank of the first error detected since last cleared.

Table 8-45: ERR2MISC1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR2STATUS.MV is set and cleared when ERR2STATUS.MV bit is cleared.
30:28	RW	CID	Indicates which logical rank on a multi-rank device, such as 3DS TSV stacks, has the error.
27:4	RO	RSVD	Reserved.
3:0	RW	BANK	Indicates the bank of the last error detected since last cleared.

8.5.3.25 Error Record 2 Miscellaneous Register 2 (ERR1MISC2)

ERR2MISC2 provides the count in the DRAM UE counters for Rank 1 and Rank 0.

Table 8-46: ERR2MISC2 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:16	RW	RANK1 _CT	The value of the Rank 1 UE counter.
15:0	RW	RANK0_CT	The value of the Rank 0 UE counter.

8.5.3.26 Error Record 2 Miscellaneous Register 3 (ERR1MISC3)

ERR2MISC3 provides the count in the DRAM UE counters for Rank 3 and Rank 2.

Table 8-47: ERR2MISC3 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:16	RW	RANK3 _CT	The value of the Rank 3 UE counter.
15:0	RW	RANK2_CT	The value of the Rank 2 UE counter.

8.5.3.27 Error Record 2 Miscellaneous Register 4 (ERR1MISC4)

ERR2MISC4 provides the count in the DRAM UE counters for Rank 5 and Rank 4.

Table 8-48: ERR2MISC4 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:16	RW	RANK5 _CT	The value of the Rank 5 UE counter.
15:0	RW	RANK4_CT	The value of the Rank 4 UE counter.

8.5.3.28 Error Record 2 Miscellaneous Register 5 (ERR1MISC5)

ERR2MISC5 provides the count in the DRAM UE counters for Rank 7 and Rank 6.

Table 8-49: ERR2MISC4 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:16	RW	RANK7 _CT	The value of the Rank 7 UE counter.
15:0	RW	RANK6_CT	The value of the Rank 6 UE counter.

8.5.3.29 Error Record 3 Feature Register (ERR3FR)

Reading ERR3FR returns the same value programmed in ERR0FR.

ERR3FR is RO.

Table 8-50 summarizes ERR3FR; the Description column provides default field values.

Table 8-50: ERR3FR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:12	RO	RSVD	Reserved.
11:10	RO	CFI	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	DE	Indicates whether errors on writes are deferred. 10: The feature is controllable.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.3.30 Error Record 3 Control Register (ERR3CTLR)

ERR3CTLR is reserved. ERR0CTLR controls this register. Reading this register returns all zeros.

Table 8-51: ERR3CTLR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.31 Error Record 3 Primary Status Register (ERR3STATUS)

ERR3STATUS reports error type, status, and contains valid bits for extra syndrome registers.

Table 8-52: ERR3STATUS Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. The ERR3ADDR register for this record contains a PA associated with the highest priority recorded error.
30	RW	V	This register is valid. At least one error was recorded.
29:28	RO	RSVD	Reserved.
27	RW	OF	This bit is directly tied to the CE counter(s) of this record. This field cannot be directly cleared. To clear this bit you must ensure all counter OF bits for this record are clear. 0: No CE counter overflowed. 1: At least one CE counter overflowed.
26:24	RO	RSVD	Reserved.
23	RW	DE	An uncorrectable DBP error was detected and deferred. This bit is always set when an DBP error is detected. If the MCU cannot defer the error, an Error Recover interrupt is also generated and recorded in the MCU Error Recovery record.
22	RO	RSVD	Reserved.
15:8	RW	IERR	Indicates which rank counter overflowed. Bit 0 is the indicator for rank 0, bit 1 is the indicator for rank 1, and so on. This field is set when an error is corrected and the corresponding counter overflows.
19:5	RO	RSVD	Reserved.
7:0	RW	SERR	Architecturally-defined primary error code; indicates the type of error. 0b0000 0000: None; no error. 0b0000 1100: ECC; error in SDRAM.

8.5.3.32 Error Record 3 Address Register 0 (ERR3ADDR0)

ERR3ADDR0 contains the physical bank, row, and column. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-53: ERR3ADDR0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:28	RW	BANK	Physical bank.
27:10	RW	ROW	Physical row.
9:0	RW	COL	Physical column.

8.5.3.33 Error Record 3 Address Register 1 (ERR3ADDR1)

ERR3ADDR0 Contains the physical CID and rank. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-54: ERR3ADDR1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:6	RO	RSVD	Reserved.
5:3	RW	CID	Physical CID.
2:0	RW	RANK	Physical rank.

8.5.3.34 Error Record 3 Miscellaneous Register 0 (ERR3MISC0)

ERR3MISC0 is reserved.

Table 8-55: ERR3MISC0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.35 Error Record 3 Miscellaneous Register 1 (ERR3MISC1)

ERR3MISC1 is reserved.

Table 8-56: ERR3MISC1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.36 Error Record 4 Feature Register (ERR4FR)

Reading ERR4FR returns the same value programmed in ERR0FR.

ERR4FR is RO.

Table 8-57 summarizes ERR4FR; the Description column provides default field values.

Table 8-57: ERR4FR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:12	RO	RSVD	Reserved.
11:10	RO	CFI	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	DE	Indicates whether errors on writes are deferred. 10: The feature is controllable.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.3.37 Error Record 4 Control Register (ERR4CTLR)

ERR4CTLR is reserved. ERR0CTLR controls this register. Reading this register returns all zeros.

Table 8-58: ERR4CTLR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.38 Error Record 4 Primary Status Register (ERR4STATUS)

ERR4STATUS reports error type, status, and contains valid bits for extra syndrome registers.

Table 8-59: ERR4STATUS Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. The ERR4ADDR register for this record contains a PA associated with the highest priority error recorded by this record.
30	RW	V	This register is valid. At least one error was recorded.
29:28	RO	RSVD	Reserved.
27	RW	OF	Overflow. Multiple errors recorded. 0: The CE counter did not overflow. 1: The CE counter overflowed.
26	RW	MV	Miscellaneous Registers Valid. The ERR4MISC registers for this record contain more information for an error recorded by this record.
25:24	RW	CE	CE. Indicates whether CEs were recorded. CE[1] is the corrected fault handling interrupt status bit. CE[1] is set when an error is corrected and the corresponding counter overflows. CE[0] does not interrupt. It is set when the counter advances and indicates new syndrome information was captured. 00: None No CEs were recorded. 01: Transient A non-overflowing error was detected and the syndrome for the error was captured. 10: Reserved 11: Persistent An error counter belonging to this record overflowed, asserting a CFH interrupt if enabled. No further syndrome information is captured until CE[1] is cleared. Check the IERR status bits to determine which counter overflowed.
23	RW	DE	An uncorrectable DBP error was detected and deferred. This bit is always set when an DBP error is detected. If the MCU cannot defer the error, an Error Recover interrupt is also generated and recorded in the MCU Error Recovery record.
22:8	RO	RSVD	Reserved.
7:0	RW	SERR	Architecturally-defined primary error code; indicates the type of error. 0b0000 0000: None (no error). 0b0000 1100: SRAM; ECC error in SRAM.

8.5.3.39 Error Record 4 Address Register 0 (ERR4ADDR0)

ERR4ADDR0 contains the physical bank, row, and column. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-60: ERR4ADDR0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:28	RW	BANK	Physical bank.
27:10	RW	ROW	Physical row.
9:0	RW	COL	Physical column.

8.5.3.40 Error Record 4 Address Register 1 (ERR4ADDR1)

ERR4ADDR0 Contains the physical CID and rank. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-61: ERR4ADDR1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:6	RO	RSVD	Reserved.
5:3	RW	CID	Physical CID.
2:0	RW	RANK	Physical rank.

8.5.3.41 Error Record 4 Miscellaneous Register 0 (ERR4MISC0)

ERR4MISC0 provides the physical rank, row, and column of the last error detected before an interrupt is asserted.

Table 8-62: ERR4MISC0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR4STATUS.MV is set and cleared when ERR4STATUS.MV is cleared.
30:28	RW	RANK	Physical rank of the last error detected before an interrupt is asserted.
27:10	RW	ROW	Physical row of the last error detected before an interrupt is asserted.
9:0	RW	COL	Physical column of the last error detected before an interrupt is asserted.

8.5.3.42 Error Record 4 Miscellaneous Register 1 (ERR4MISC1)

ERR4MISC1 provides the bank, logical rank, and Data Buffer ID (DBID) of the last error detected before an interrupt is asserted.

Table 8-63: ERR4MISC1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR4STATUS.MV is set and cleared when ERR4STATUS.MV bit is cleared.
30:28	RW	CID	Indicates which logical rank on a multi-rank device, such as 3DS TSV stacks, has the error.
27:11	RO	RSVD	Reserved.
10:4	RW	DBID	DBID; the buffer entry (0 to 127) in which the error occurred. Failed buffers can be excluded from allocation by using this DBID information to program the queue_allocate_control registers. This eliminates the chance of subsequent errors caused by a failed buffer.
3:0	RW	BANK	Indicates the bank of the last error detected before an interrupt is asserted.

8.5.3.43 Error Record 5 Feature Register (ERR5FR)

Reading ERR5FR returns the same value programmed in ERR0FR.

ERR5FR is RO.

Table 8-64 summarizes ERR5FR; the Description column provides default field values.

Table 8-64: ERR5FR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:12	RO	RSVD	Reserved.
11:10	RO	CFI	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	DE	Indicates whether errors on writes are deferred. 10: The feature is controllable.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.3.44 Error Record 5 Control Register (ERR5CTLR)

ERR4CTLR is reserved. ERR0CTLR controls this register. Reading this register returns all zeros.

Table 8-65: ERR5CTLR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.45 Error Record 5 Primary Status Register (ERR5STATUS)

ERR5STATUS reports error type, status, and contains valid bits for extra syndrome registers.

Table 8-66: ERR5STATUS Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. The ERR5ADDR register for this record contains a PA associated with the highest priority error recorded by this record.
30	RW	V	This register is valid. At least one error was recorded.
29:28	RO	RSVD	Reserved.
27	RW	OF	Overflow. Multiple errors recorded. 0: The CE counter did not overflow. 1: The CE counter overflowed.
26	RW	MV	Miscellaneous Registers Valid. The ERR5MISC registers for this record contain additional information for an error recorded by this record.
25:24	RO	RSVD	Reserved.
23	RW	DE	An uncorrectable ECC error was detected and deferred. This bit is always set when an uncorrectable ECC error is detected.
22:8	RO	RSVD	Reserved.
7:0	RW	SERR	Architecturally-defined primary error code; indicates the type of error. 0b0000 0000: None (no error). 0b0000 1100: SRAM; ECC error in SRAM.

8.5.3.46 Error Record 5 Address Register 0 (ERR5ADDR0)

ERR5ADDR0 contains the physical bank, row, and column. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-67: ERR5ADDR0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:28	RW	BANK	Physical bank.
27:10	RW	ROW	Physical row.
9:0	RW	COL	Physical column.

8.5.3.47 Error Record 5 Address Register 1 (ERR5ADDR1)

ERR5ADDR0 Contains the physical CID and rank. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-68: ERR5ADDR1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:6	RO	RSVD	Reserved.
5:3	RW	CID	Physical CID.
2:0	RW	RANK	Physical rank.

8.5.3.48 Error Record 5 Miscellaneous Register 0 (ERR5MISC0)

ERR5MISC0 provides the physical rank, row, and column of the last error detected before an interrupt is asserted.

Table 8-69: ERR5MISC0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR5STATUS.MV is set and cleared when ERR5STATUS.MV bit is cleared.
30:28	RW	RANK	Physical rank of the last error detected before an interrupt is asserted.
27:10	RW	ROW	Physical row of the last error detected before an interrupt is asserted.
9:0	RW	COL	Physical column of the last error detected before an interrupt is asserted.

8.5.3.49 Error Record 5 Miscellaneous Register 1 (ERR5MISC1)

ERR5MISC1 provides the bank, logical rank, and DBID of the last error detected before an interrupt is asserted.

Table 8-70: ERR5MISC1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR5STATUS.MV is set and cleared when ERR5STATUS.MV is cleared.
30:28	RW	CID	Indicates which logical rank on a multi-rank device, such as 3DS TSV stacks, has the error.
27:11	RO	RSVD	Reserved.
10:4	RW	DBID	DBID; the buffer entry (0 to 127) in which the error occurred. Failed buffers can be excluded from allocation by using this DBID information to program the queue_allocate_control registers. This eliminates the chance of subsequent errors caused by a failed buffer.
3:0	RW	BANK	Indicates the bank of the last error detected before an interrupt is asserted.

8.5.3.50 Error Record 6 Feature Register (ERR6FR)

Reading ERR6FR returns the same value programmed in ERR0FR.

ERR6FR is RO.

Table 8-71 summarizes ERR6FR; the Description column provides default field values.

Table 8-71: ERR6FR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:12	RO	RSVD	Reserved.
11:10	RO	CFI	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	DE	Indicates whether errors on writes are deferred. 10: The feature is controllable.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.3.51 Error Record 6 Control Register (ERR6CTLR)

ERR4CTLR is reserved. ERR0CTLR controls this register. Reading this register returns all zeros.

Table 8-72: ERR6CTLR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.3.52 Error Record 6 Primary Status Register (ERR6STATUS)

ERR6STATUS reports error type, status, and contains valid bits for extra syndrome registers.

Table 8-73: ERR6STATUS Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. The ERR6ADDR register for this record contains a PA associated with the highest priority error.
30	RW	V	This register is valid. At least one error was recorded.
29	RW	UE	UE. At least one error was detected that was not corrected.
28	RW	ER	Error Reported. A UE was reported to a primary device.
27	RW	OF	Overflow. Multiple errors recorded. 0: Only one error described by ERR6STATUS.UE was detected. 1: At least one error described by ERR6STATUS.UE was discarded.
26	RW	MV	Miscellaneous Registers Valid. The ERR6MISC registers for this record contain more information for a recorded.
25:23	RO	RSVD	Reserved.
22	RW	PN	Poison Detected. This bit is set when an uncorrectable ECC error cannot be deferred.
21:20	RW	UICT	Uninfected and Containable. Describes the state of the component after detecting a UE. A UE is neither corrected nor deferred. A DE is read data which was poisoned and not flagged as an error to the primary device. 00: No UE detected 11: UER; Recoverable (uninfected and containable) error. A UE or poisoned data was detected when reading DRAM.
19:8	RO	RSVD	Reserved.
7:0	RW	SERR	Architecturally-defined primary error code; indicates the type of error. 0b0000 0000: None (no error). 0b0000 1100: SRAM; ECC error in SRAM.

8.5.3.53 Error Record 6 Address Register 0 (ERR6ADDR0)

ERR6ADDR0 contains the physical bank, row, and column. If an error is associated with a PA, the PA must be written to the address register when the error is recorded. .

Table 8-74: ERR6ADDR0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:28	RW	BANK	Physical bank.
27:10	RW	ROW	Physical row.
9:0	RW	COL	Physical column.

8.5.3.54 Error Record 6 Address Register 1 (ERR6ADDR1)

ERR6ADDR0 Contains the physical CID and rank. If an error is associated with a PA, the PA must be written to the address register when the error is recorded.

Table 8-75: ERR6ADDR1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:6	RO	RSVD	Reserved.
5:3	RW	CID	Physical CID.
2:0	RW	RANK	Physical rank.

8.5.3.55 Error Record 6 Miscellaneous Register 0 (ERR6MISC0)

ERR6MISC0 provides the physical rank, row, and column of the last error detected before an interrupt is asserted.

Table 8-76: ERR6MISC0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR6STATUS.MV is set and cleared when ERR6STATUS.MV bit is cleared.
30:28	RW	RANK	Physical rank of the last error detected before an interrupt is asserted.
27:10	RW	ROW	Physical row of the last error detected before an interrupt is asserted.
9:0	RW	COL	Physical column of the last error detected before an interrupt is asserted.

8.5.3.56 Error Record 6 Miscellaneous Register 1 (ERR6MISC1)

ERR6MISC1 provides the bank, logical rank, and DBID of the last error detected before an interrupt is asserted.

Table 8-77: ERR6MISC1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	VALID	The register contains valid syndrome information. The bit is set when ERR6STATUS.MV is set and cleared when ERR6STATUS.MV bit is cleared.
30:28	RW	CID	Indicates which logical rank on a multi-rank device, such as 3DS TSV stacks, has the error.
27:11	RO	RSVD	Reserved.
10:4	RW	DBID	DBID; the buffer entry (0 to 127) in which the error occurred. Failed buffers can be excluded from allocation by using this DBID information to program the queue_allocate_control registers. This could eliminate the chance of subsequent errors caused by a failed buffer.
3:0	RW	BANK	Indicates the bank of the last error detected before an interrupt is asserted.

8.5.3.57 Error Record Group Status Register (ERRGSR)

This register shows the status of the MCU error record status registers. When a CFH, FH, or ER interrupt occurs, software can check this register to tell which error record or records caused the interrupt.

Each RW bit in this register contains a copy of the V bit of the corresponding error record status register.

Table 8-78: ERRGSR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:7	RO	RSVD	Reserved.
6	RW	S6	ERR6STATUS.V value. 0: No error. 1: One or more errors.
5	RW	S5	ERR5STATUS.V value. 0: No error. 1: One or more errors.
4	RW	S4	ERR4STATUS.V value. 0: No error. 1: One or more errors.
3	RW	S3	ERR3STATUS.V value. 0: No error. 1: One or more errors.

Table 8-78: ERRGSR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
2	RW	S2	ERR2STATUS.V value. 0: No error. 1: One or more errors.
1	RW	S1	ERR1STATUS.V value. 0: No error. 1: One or more errors.
0	RW	S0	ERR0STATUS.V value. 0: No error. 1: One or more errors.

8.5.3.58 Link Error Count Register (link_err_count)

This register maintains a count of link errors. The counter increments on detection of a new link error (dfi_alert_n or dfi_err).

Table 8-79: link_err_count Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:16	RO	RSVD	Reserved.
15:0	RW	link_err_count	Link error count. A write to this field resets the counter to the written value.

8.5.3.59 Link Error Interrupt Access Information Register 31:0 (link_err_int_info_31_00)

This register provides information about interrupt access restrictions.

Table 8-80: link_err_int_info_31_00 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:7	RO	RSVD	Reserved.
6	RW	link_err_int_info_alert_n	link_err_int_info_alert_n. Returns the value sent on the dfi_alert_n signal.
5	RW	link_err_int_info_err	Returns the value sent on the dfi_err signal.
4:1	RW	link_err_int_info	For a dfi_err error, returns the corresponding value sent on dfi_err_info. For all other errors it is not used
0	RW	link_err_int_info_30_00_valid	Indicates whether an interrupt (single interrupt or multiple interrupt) occurred and whether the interrupt information is valid. A one indicates it is valid and that no overflow occurred. If link_err_int_info_err and link_err_int_info_alert_n are not asserted, this indicates data_capture error.

8.5.3.60 Link Error Interrupt Access Information Register 63:32 (link_err_int_info_63_32)

This register provides information about interrupt access restrictions.

Table 8-81: link_err_int_info_63_32 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31	RW	link_err_int_info_61_31_valid	Indicates whether an interrupt (single interrupt or multiple interrupt) occurred and whether the interrupt information is valid. A one indicates it is valid and that no overflow occurred.
30:0	RO	RSVD	Reserved.

8.5.4 PCI Express (PCIe) Host Bridge Error Record Registers

Table 8-82: PCIe Error Record Registers

REGISTER	OFFSET	ACCESS	SIZE	DESCRIPTION
ERR0FR	0xF000	RW	32-bit	Error Record 0 Feature Register
ERR0CTLR	0xF008	RO	32-bit	Error Record 0 Control Register
ERR0STATUS	0xF010	RW	32-bit	Error Record 0 Status Register
ERR0ADDR0	0xF018	RO	32-bit	Error Record 0 Address Register 0
ERR0ADDR1	0xF01C	RW	32-bit	Error Record 0 Address Register 1
ERR0MISC0	0xF020	RW	32-bit	Error Record 0 Miscellaneous Register 0
ERR0MISC1	0xF028	RO	32-bit	Error Record 0 Miscellaneous Register 1
ERRGEN	0xFE08	RW	32-bit	Error Generation Register

8.5.4.1 Error Record 0 Feature Register (ERR0FR)

ERR0FR defines features which are common to all RAS error records in this section. Each field defines which architecturally-defined common features are implemented and, of the implemented features, which are software programmable.

ERR0FR is RO.

Table 8-83 summarizes ERR0FR; the Description column provides default field values.

Table 8-83: ERR0FR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:10	RO	RSVD	Reserved.
11:10	RO	CFI	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	RSVD	Reserved.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.4.2 Error Record 0 Control Register (ERR0CTLR)

ERR0CTLR contains a bit for enabling and disabling an error recovery interrupt.

Table 8-84: ERR0CTLR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:3	RO	RSVD	Reserved.
2	RW	UI	Error recovery interrupt enable; serves as the interrupt mask for the Host Bridge UE (HB_UERR) interrupt. 0: Error recovery disabled. 1: Error recovery interrupt enabled.
1:0	RO	RSVD	Reserved.

8.5.4.3 Error Record 0 Primary Status Register (ERR0STATUS)

ERR0STATUS contains information about the error record.

Table 8-85: ERR0STATUS Register Format (Sheet 1 of 3)

BIT	TYPE	FIELD	DESCRIPTION
31	RW1C	AV	Address Valid. 0: ERR0ADDR is not valid. 1: ERR0ADDR contains an address associated with the highest priority error recorded by this record. This bit ignores writes if ERR0STATUS.{CE, DE, UE} are set to 1, and the highest priority of these is not cleared in the same write. This bit is Read/Write-One-to-Clear (RW1C).
30	RW1C	V	Status Register valid. The possible values are: 0: ERR0STATUS is not valid. 1: ERR0STATUS is valid. At least one error was recorded. This bit ignores writes if Armv8.4-RAS is not implemented and any of ERR0STATUS.{CE, DE, UE} are set to 1, and the highest priority of these is not cleared to 0 in the same write. This bit is RW1C.
29	RW1C	UE	UE. 0: No error that cannot be corrected or deferred was detected. 1: At least one error that could not be corrected or deferred was detected. If error recovery interrupts are enabled, the interrupt signal is asserted until this bit is cleared. This bit is RW1C.

Table 8-85: ERR0STATUS Register Format (Sheet 2 of 3)

BIT	TYPE	FIELD	DESCRIPTION
28	RW	ER	Error reported. 0: No external abort was reported. 1: The node reported an external abort to the primary device in access or making a transaction. If this bit is 1, software must write a 1 to this bit to clear it to 0 when: • Clearing ERR0STATUS.V to 0. • Clearing ERR0STATUS.UE to 0, This bit ignores writes if any of ERR0STATUS.{CE, DE, UE} are set to 1, and the highest priority of these is not cleared to 0 in the same write. This bit is not valid and reads UNKNOWN if any of these are true: • ERR0STATUS.V is set to 0. • ERR0STATUS.UE is set to 0 and this bit is never set to 1 by a DE. • ERR0STATUS.{UE, DE} are both set to 0 and this bit can be set to 1 by a DE. This bit drives HB_UERR interrupt. This bit is RW1C.
27	RW1C	OF	Overflow. 0: No error syndrome was discarded. 1: More than one error detected; this bit is set when any of these occur: • A CE counter is implemented, an error is counted, and the counter overflows. • ERR0STATUS.V was previously set to 1, a CE counter is not implemented, and a CE is recorded. • ERR0STATUS.V was previously set to 1, and a type of error other than a CE is recorded. If this bit is 1, software must write 1 to it to clear this bit to 0 when clearing ERR0STATUS.V to 0. This bit is RW1C.
26	RW	MV	Miscellaneous Registers Valid. 0: ERR0MISC0 and ERR0MISC1 are not valid. 1: ERR0MISC0 and ERRMISC0 Contain more information about any error recorded in this record. This bit is RW1C.
25:24	RO	RSVD	Reserved.
23	RW1C	DE	DE. 0: No errors were deferred. 1: At least one error was not corrected and deferred by poisoning. This bit is RW1C.

Table 8-85: ERR0STATUS Register Format (Sheet 3 of 3)

BIT	TYPE	FIELD	DESCRIPTION
22	RW1C	PN	Poison. 0: UE or DE recorded because a corrupt value was detected, for example, by an Error Detection Code (EDC). 1: UE or DE recorded because a poison value was detected. If Armv8.4-RAS is implemented and this bit is 1, software must write 1 to this bit to clear this bit to 0 when clearing ERR0STATUS.V to 0. This bit is RW1C.
21:16	RO	RSVD	Reserved.
15:8	RW	IERR	The subset of architecturally-defined values that this field can take is implementation-defined. If any value not in this set is written to this field, the value read back from the field is UNKNOWN. This field ignores writes if any of ERR0STATUS.{CE, DE, UE} are set to 1, and the highest priority of these is not being cleared to 0 in the same write. 0b0000 0000: No Error. 0b0000 0001: Host bridge data parity error. 0b0000 0010: Poisoned completion from Endpoint (EP). 0b0000 0011: Host bridge write poison data. This field is not valid and reads UNKNOWN if ERR0STATUS.V is set to 0.
7:0	RW	SERR	Architecturally-defined primary error code. Indicates the type of error: 0: No error. 1: Implementation-defined error. 2 Data value from (non-associative) internal memory which encountered an ECC single-bit error and could correct it. 10: Data value from producer, for example, poison on the write data bus. This field ignores writes if any of ERR0STATUS.{CE, DE, UE} are set to 1, and the highest priority of these is not cleared to 0 in the same write. This field is not valid and reads UNKNOWN if ERR0STATUS.V is set to 0.

8.5.4.4 Error Record 0 Address Register 0 (ERR0ADDR0)

Table 8-86: ERR0ADDR0 Register Format

BIT	TYPE	FIELD	DEFAULT	DESCRIPTION
31:0	RW	PADDR [31:0]	0x0000 0000	PA [31:0].

8.5.4.5 Error Record 0 Address Register 1 (ERR0ADDR1)

Table 8-87: ERR0ADDR1 Register Format

BIT	TYPE	FIELD	DEFAULT	DESCRIPTION
31:0	RW	PADDR [55:32]	0x0000 0000	PA [55:32].

8.5.4.6 Error Record 0 Miscellaneous Register 0 (ERR0MISC0)

Table 8-88: ERR0MISC0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:2	RO	RSVD	Reserved.
1	RW	REQSEC	NS bit for transaction. 0: Secure world access. 1: Normal world access.
0	RW	RQTYPE	Request type. 0: Write. 1: Read.

8.5.4.7 Error Record 0 Miscellaneous Register 1 (ERR0MISC1)

ERR0MISC1 is reserved.

Table 8-89: ERR0MISC1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.4.8 Error Generation Register (ERRGEN)

Table 8-90: ERRGEN Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:2	RO	RSVD	Reserved.
1	RW, W1S, SC	GEN_UE	Generate a Host Bridge data check parity UE. This bit is Write-One-to-Set (W1S) and Self-Clear (SC).
0	RO	RSVD	Reserved.

8.5.5 Generic Interrupt Controller (GIC) Error Record Registers

The GIC has several error records.

Table 8-91: GIC Error Records

RECORD	DESCRIPTION
0	Uncorrected software error in the Distributor.
1	Corrected SPI RAM error.
2	Uncorrected SPI RAM error.
3	Corrected SGI RAM error.
4	Uncorrected SGI RAM error.
5	Reserved.
6	Reserved.
7	Corrected PPI RAM error.
8	Uncorrected PPI RAM error.
9	Corrected LPI RAM error.
10	Uncorrected LPI RAM error.
11	Corrected ITS RAM error.
12	Uncorrected ITS RAM error.
13+	Uncorrected software error in ITS; one record per ITS on the processor.

In the error record register names, *n* refers to a GIC error record listed in *Table 8-91*.

Table 8-92: GIC Error Record Registers (Sheet 1 of 2)

REGISTER	OFFSET	ACCESS	SIZE	DESCRIPTION
GICT_ERR*n*FR	0x0000 + (*n* × 0x40)	RO	64-bit	GICT Error Record *n* Feature Register
GICT_ERR*n*CTLR	0x0008 + (*n* × 0x40)	RW	64-bit	GICT Error Record *n* Control Register
GICT_ERR*n*STATUS	0x0010 + (*n* × 0x40)	RW	32-bit	GICT Error Record *n* Primary Status Register
GICT_ERR*n*ADDR	0x0018 + (*n* × 0x40)	RW	64-bit	GICT Error Record *n* Address Register
GICT_ERR*n*MISC0	0x0020 + (*n* × 0x40)	RW	64-bit	GICT Error Record *n* Miscellaneous Register 0
GICT_ERR*n*MISC1	0x0038 + (*n* × 0x40)	RW	64-bit	GICT Error Record *n* Miscellaneous Register 1
GICT_ERRGSR	0xE000	RO	64-bit	GICT Error Record Group Status Register
GICT_ERRIRQCRn	0xE800 – 0xE808	RW	32-bit	GICT Error Interrupt Configuration Registers

Table 8-92: GIC Error Record Registers (Sheet 2 of 2)

REGISTER	OFFSET	ACCESS	SIZE	DESCRIPTION
GICT_DEVARCH	0xFFBC	RO	32-bit	GICT Device Architecture Register
GICT_ERRIDR	0xFFC8	RO	32-bit	GICT Error Record ID Register
GICT_PIDR4	0xFFD0	RO	32-bit	GICT Peripheral ID Register 4
GICT_PIDR5	0xFFD4	RO	32-bit	GICT Peripheral ID Register 5
GICT_PIDR6	0xFFD8	RO	32-bit	GICT Peripheral ID Register 6
GICT_PIDR7	0xFFDC	RO	32-bit	GICT Peripheral ID Register 7
GICT_PIDR0	0xFFE0	RO	32-bit	GICT Peripheral ID Register 0
GICT_PIDR1	0xFFE4	RO	32-bit	GICT Peripheral ID Register 1
GICT_PIDR2	0xFFE8	RO	32-bit	GICT Peripheral ID Register 2
GICT_PIDR3	0xFFEC	RO	32-bit	GICT Peripheral ID Register 3
GICT_CIDR0	0xFFF0	RO	32-bit	GICT Component ID Register 0
GICT_CIDR1	0xFFF4	RO	32-bit	GICT Component ID Register 1
GICT_CIDR2	0xFFF8	RO	32-bit	GICT Component ID Register 2
GICT_CIDR3	0xFFFC	RO	32-bit	GICT Component ID Register 3

8.5.5.1 GICT Error Record n Feature Register (GICT_ERRnFR)

GICT_ERRnFR defines features which are common to all RAS error records in this section. Each field defines which architecturally-defined common features are implemented and, of the implemented features, which are software programmable.

GICT_ERRnFR is RO.

Table 8-93 summarizes GICT_ERRnFR; the Description column provides default field values.

Table 8-93: GICT_ERR0FR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
63:16	RO	RSVD	Reserved.
15	RO	RP	Indicates whether the corrected fault handling interrupt is enabled. 10: The feature is controllable.
14:12	RO	CEC	CE Counter. 010: The node implements an 8-bit standard CE counter in GICT_ERRnMISC0[39:32]

Table 8-93: GICT_ERR0FR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
11:10	RO	CFI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
9:8	RO	UE	Indicates whether UE reporting is enabled. 01: The feature is always enabled.
7:6	RO	FI	Indicates whether the fault handling interrupt is enabled. 10: The feature is controllable.
5:4	RO	UI	Indicates whether the uncorrected error recovery interrupt is enabled. 10: The feature is controllable.
3:2	RO	DE	Indicates whether errors on writes are deferred. 10: The feature is controllable.
1:0	RO	ED	Indicates whether error detection and correction is enabled. 10: The feature is always enabled.

8.5.5.2 GICT Error Record n Control Register (GICT_ERRnCR)

Table 8-94: GICT_ERRnCR Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
63:16	RO	RSVD	Reserved (RAZ).
15	RW	RP	0: An error response to a transaction is reported.
14:9	RO	RSVD	Reserved (RAZ).
8	RW	CFI	Controls whether a CE generates a fault handling interrupt. SBZ on UEs, otherwise: 0: The GIC does not assert a fault handling interrupt for CEs. 1: The GIC asserts a fault handling interrupt when a CE occurs.
7:5	RO	RSVD	Reserved (RAZ).
4	RW	UE	UE. RAZ/WI for all records except GICT Error Record (0), otherwise: 0: Do not send external abort with transaction. 1: Send external abort with transaction.
3	RW	FI	Fault handling interrupt. SBZ on CE records, otherwise: 0: Fault handling interrupt is not generated on any error. 1: Fault handling interrupt is generated on all UEs.

Table 8-94: GICT_ERRnCR Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
2	RW	UE	Error recovery interrupt for UE. SBZ on CE records, otherwise: 0: Error recovery interrupt is not generated on any error. 1: Error recovery interrupt is generated on all UEs.
1:0	RO	RSVD	Reserved (RAZ).

8.5.5.3 GICT Error Record n Primary Status Register (GICT_ERRnSTATUS)

Table 8-95: GICT_ERRnSTATUS Register Format (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Indicates whether the address is valid. 0: GICT_ERRnADDR is not valid. 1: GICT_ERRnADDR contains an address associated with the highest priority error stored by this record; present only in record 0.
30	RW	V	Indicates whether this register is valid. 0: GICT_ERRnSTATUS is not valid. 1: GICT_ERRnSTATUS is valid; one or more errors are recorded.
29	RW	UE	UE bit. SBZ in CE records.
28	RW	ER	Indicates that at least one error is been reported over ACE-Lite. Set for record 0 only, and only for accesses to corrupted data, and bad incoming access.
27	RW	OF	The record overflowed.
26	RW	MV	Indicates whether the GICT miscellaneous registers are valid. 0: GICT_ERRnMISC0 and GICT_ERRnMISC1 are not valid. 1: GICT_ERRnMISC0 and GICT_ERRnMISC1 are valid.
25:24	RW	CE	CE; indicates errors that are correctable: 0b00: No CE recorded. 0b10: At least one CE recorded.
23:22	RO	RSVD	Reserved (RAZ).
21:20	RW	UET	UE Type. 00: Uncontainable.
19:16	RO	RSVD	Reserved (RAZ).

Table 8-95: GICT_ERRnSTATUS Register Format (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
15:8	RW	IERR	This field is RO in Record 0 apart from Record 0.
7:0	RW	SERR	This field is RO in Record 0 apart from Record 0 and Record 13 (and above).

8.5.5.4 GICT Error Record n Address Register (GICT_ERRnADDR)

Table 8-96: GICT_ERRnADDR Register Format

BIT	TYPE	FIELD	DESCRIPTION
63	RW	NS	Non-secure attribute: 0: The address is Secure. 1: The address is Non-secure.
62:48	RO	RSVD	Reserved (RAZ/WI).
47:0	RW	PADDR	The PA of the error.

8.5.5.5 GICT Error Record n Miscellaneous Register 0 (GICT_ERRnMISC0)

Table 8-97: GICT_ERRnMISC0 Register Format

BIT	TYPE	FIELD	DESCRIPTION
63:42	RO	RSVD	Reserved (RAZ).
41	RW	RE	Rounding Error (RE). The RE counter is underreporting.
40	RW	OF	Sticky overflow bit. 0: The counter has not overflowed. 1: The counter has overflowed. If the corrected fault handling interrupt is enabled, the GIC generates a fault handling interrupt.
39:32	RW	CE	CE count. The CE counter is not 0 or is more than 134; incremented for each CE that does not match the recorded syndrome.
31:0	RW	DATA	Information associated with the error.

8.5.5.6 GICT Error Record n Miscellaneous Register 1 (GICT_ERRnMISC1)

This register contains the data value of a UE in the LPI RAM, or ITS software information for one of 13 or more, error records. The GIC supports a single GICT_ERRnMISC1 register.

Table 8-98: GICT_ERRnMISC1 Register Format

BIT	TYPE	FIELD	DESCRIPTION
63:x + 1	RO	RSVD	Reserved (RAZ).
x:0	RW	INFO	This value represents either data written to LPI RAM when a UE is detected, or ITS software information for one of 13 or more error records. The value of x depends upon the width of the LPI RAM, which is set during GIC configuration.

8.5.5.7 GICT Error Record Group Status Register (GICT_ERRGSR)

Table 8-99: GICT_ERRGSR Register Format

BIT	TYPE	FIELD	DESCRIPTION
n	RW	STATUS	Indicates the status of GIC error record *n*, where *n* is 0 - 13+ depending upon the configuration (see *Table 8-91*). The bit value also corresponds to *n*. 0: The error record is not reporting any errors. 1: The error record is reporting one or more errors.

8.5.5.8 Error Interrupt Configuration Registers (GICT_IRQCRn)

GICT_ERRIRQCR0 controls the fault handling interrupts. GICT_ERRIRQCR0 Controls the error recovery interrupts.

Table 8-100: GICT_IRQCRn Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:10	RO	RSVD	Reserved (RAZ).
9:0	RW	SPIID	SPI ID. Returns 0 if an invalid entry is written. In a 2P system, the SPIID field must be programmed only to an SPI ID that the chip owns.

8.5.5.9 Error Record ID Register (GICT_ERRIDR)

Table 8-101: GICT_ERRIDR Register Format

BIT	TYPE	FIELD	DESCRIPTION
31:16	RO	RSVD	Reserved (RAZ).
15:0	RO	NUM	Identifies the device configuration. 10: No LPI available. 12: LPI is available but no ITS. 14: LPI is available and 1 × ITS. 15: LPI is available and 2 × ITS. 16: LPI is available and 3 × ITS.

8.5.6 Ampere Link Interconnect (ALI) Error Record Registers

Table 8-102: ALI Error Record Registers (Sheet 1 of 2)

REGISTER	ACCESS	SIZE	DESCRIPTION
ERR0FR	RO	32	Error Record 0 Feature Register.
ERR0CTLR	RW	32	Error Record 0 Control Register.
ERR0STATUS	RW	32	Error Record 0 Primary Status Register.
ERR0ADDR	RW	32	Error Record 0 Address Register.
ERR0MISC0	RW	64	Error Record 0 Miscellaneous Register 0.
ERR0MISC1	RO	64	Error Record 0 Miscellaneous Register 1.
ERR1FR	RO	32	Error Record 1 Feature Register.
ERR1CTLR	RW	32	Error Record 1 Control Register.
ERR1STATUS	RW	32	Error Record 1 Primary Status Register.
ERR1ADDR	RW	32	Error Record 1 Address Register.
ERR1MISC0	RW	64	Error Record 1 Miscellaneous Register 0.
ERR1MISC1	RO	64	Error Record 1 Miscellaneous Register 1.
ERR2FR	RO	32	Error Record 2 Feature Register.
ERR2CTLR	RW	32	Error Record 2 Control Register.
ERR2STATUS	RW	32	Error Record 2 Primary Status Register.
ERR2ADDR	RW	32	Error Record 2 Address Register.
ERR2MISC0	RW	64	Error Record 2 Miscellaneous Register 0.
ERR2MISC1	RO	64	Error Record 2 Miscellaneous Register 1.
ERR3FR	RO	32	Error Record 3 Feature Register.

Table 8-102: ALI Error Record Registers (Sheet 2 of 2)

REGISTER	ACCESS	SIZE	DESCRIPTION
ERR3CTLR	RW	32	Error Record 3 Control Register.
ERR3STATUS	RW	32	Error Record 3 Primary Status Register.
ERR3ADDR	RW	32	Error Record 3 Address Register.
ERR3MISC0	RW	64	Error Record 3 Miscellaneous Register 0.
ERR3MISC1	RO	64	Error Record 3 Miscellaneous Register 1.
ERRGSR	RW	32	Error Group Status Register.

8.5.6.1 Error Record 0 Feature Register (ERR0FR)

ERR0FR defines features which are common to all RAS error records in this section. Each field defines which architecturally-defined common features are implemented and, of the implemented features, which are software programmable.

ERR0FR is RO.

Table 8-103 summarizes ERR0FR; the Description column provides default field values.

Table 8-103: Error Record 0 Feature Register (ERR0FR) (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:20	RO	RSVD	Reserved.
19:18	RW	CEO	CE overwrite. Indicates the behavior when a second CE is detected after a first CE is recorded by the node. 00: Count CE if a counter is implemented. Keep the previous error syndrome. If the counter overflows, or no counter is implemented, ERR0STATUS.OF is set to 1.
17:16	RO	DUI	Error recovery interrupt for Deferred Errors (DEs). ERR0 does not support this feature.
15	RO	RP	Repeat counter. Indicates whether the node implements a repeat CE counter. 0: One CE counter is implemented.
14:12	RO	CEC	CE counter. Indicates whether the node implements a standard CE counter in ERR0MISC0. 010: The node implements a 16-bit standard CE counter in ERR0MISC0[47:32].
11:10	RO	CFI	Fault handling interrupt for CEs. Indicates whether the node implements a control for enabling fault handling interrupts on CEs. 10: The feature is controlled using ERR0CTLR.CFI.
9:8	RO	UE	In-band Uncorrected Error (UE) reporting. ERR0 does not support this feature.

Table 8-103: Error Record 0 Feature Register (ERR0FR) (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
7:6	RO	FI	Fault handling interrupt. Indicates whether the node implements a fault handling interrupt, and, if so, whether the node implements controls for enabling and disabling it. 10: The feature is controlled using ERR0CTLR.FI.
5:4	RO	UI	Error recovery interrupt for UEs. ERR0 does not support this feature.
3:2	RO	RSVD	Reserved.
1:0	RO	ED	Error detection and correction. Indicates whether the node implements controls for enabling and disabling error reporting and logging. 10: The feature is controlled using ERR0CTLR.ED.

8.5.6.2 Error Record 0 Control Register (ERR0CTLR)

ERR0CTLR controls features such as error reporting and interrupt enables.

Table 8-104: Error Record 0 Control Register (ERR0CTLR) (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:11	RO	RSVD	Reserved.
10	R0	DUI	Error recovery interrupt for DEs enable. ERR0 does not support this feature.
9	RO	RSVD	Reserved.
8	RW	CFI	Fault handling interrupt for CEs enable. 0: Fault handling interrupt is not generated for CEs. 1: Fault handling interrupt is generated for CEs. When enabled: Because the node implements a CE counter, the fault handling interrupt is generated when the counter overflows and the counter overflow flag is set. The interrupt is generated even when the error syndrome is discarded because the error record already recorded a higher priority error.
7:5	RO	RSVD	Reserved.
4	RO	UE	In-band UE reporting enable. ERR0 does not support this feature.

Table 8-104: Error Record 0 Control Register (ERR0CTLR) (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
3	RO	CFI	Fault handling interrupt enable. 0: Fault handling interrupt is enabled. 1: Fault handling interrupt is not enabled. When enabled: Because the node implements a CE counter, the fault handling interrupt is generated when the counter overflows and the counter overflow flag is set. The interrupt is generated even when the error syndrome is discarded because the error record already recorded a higher priority error.
2	RO	UI	UE recovery interrupt enable. ERR0 does not support this feature.
1	RO	RSVD	Reserved.
0	RO	ED	Error reporting and logging enable. 0: Error reporting disabled. 1: Error reporting enabled. When disabled: The node behaves as if error detection is disabled. The node does not record or signal errors.

8.5.6.3 Error Record 0 Primary Status Register (ERR0STATUS)

ERR0STATUS contains information about the error record.

Table 8-105: Error Record 0 Primary Status Register (ERR0STATUS) (Sheet 1 of 3)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. ERR0 does not support this feature.
30	RW	V	Status Register valid. 0: ERR0STATUS is not valid. 1: ERR0STATUS is valid. At least one error is recorded. This bit ignores writes if any of ERR0STATUS.{CE, DE, UE} are set to 1 and not cleared to 0 in the same write. This bit is read/write-one-to-clear. This bit resets to zero on a Cold reset.
29	RW	UE	UE. ERR0 does not support this feature.
28	RW	ER	Error Reported. ERR0 does not support this feature.

Table 8-105: Error Record 0 Primary Status Register (ERR0STATUS) (Sheet 2 of 3)

BIT	TYPE	FIELD	DESCRIPTION
27	RW	OF	Overflow. Indicates that multiple errors are detected. This bit is set to 1 when ERR0STATUS.UE == 0, ERR0STATUS.DE == 0, and the CE counter overflows. If this bit is nonzero, software must write 1 to this bit to clear this bit to zero when clearing ERR0STATUS.V to 0. This bit is not valid and reads UNKNOWN if ERR0STATUS.V is set to 0. This bit is read/write-one-to-clear.
26	RW	MV	Miscellaneous Registers Valid. 0: ERR0MISC0 and ERR0MISC1 are not valid. 1: The IMPLEMENTATION DEFINED contents of the ERR0MISC0 and ERR0MISC1 registers contain more information for an error recorded by this record. This bit ignores writes if any of ERR0STATUS.{CE, DE, UE} are set to 1, and the highest priority of these is not cleared to 0 in the same write. This bit is read/write-one-to-clear. This bit resets to zero on a Cold reset.
25:24	RW	CE	CE. 00: No CE is recorded. 01: Not supported. 10: At least one CE is recorded. 11: Not supported. If this field is nonzero, software must write ones to this field to clear this field to zero when clearing ERR0STATUS.V to 0. This field ignores writes if ERR0STATUS.OF is set to 1 and is not cleared to 0 in the same write. This field is not valid and reads UNKNOWN if ERR0STATUS.V is set to 0. This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an UNKNOWN value.
23	RW	DE	DE. ERR1 does not support this feature.
22	RW	PN	Poison. ERR0 does not support this feature.
21:20	RW	UET	UE Type. ERR0 does not support this feature.
19:16	RO	RSVD	Reserved.
15:8	RW	IERR	Implementation-defined error code. 0b0000 0000: No error. 0b0000 0001: TX/RX Buffer Single Bit ECC Error.

Table 8-105: Error Record 0 Primary Status Register (ERR0STATUS) (Sheet 3 of 3)

BIT	TYPE	FIELD	DESCRIPTION
7:0	RW	SERR	Architecturally defined primary error code. Indicates the type of error. A fault handling agent may use the primary error code to identify an error without requiring a device-specific code, for example, to count CEs and handle thresholds in software, or generate a short log entry. 0b0000 0000: No error. 0b0000 0001: IMPLEMENTATION DEFINED error.

8.5.6.4 Error Record 0 Address Register (ERR0ADDR)

ERR0ADDR is reserved.

Table 8-106: Error Record 0 Address Register (ERR0ADDR)

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.6.5 Error Record 0 Miscellaneous Register 0 (ERR0MISC0)

ERR0MISC0 contains a CE counter and other state information not available in the corresponding status and address error record registers.

Table 8-107: Error Record 0 Miscellaneous Register 0 (ERR0MISC0)

BIT	TYPE	FIELD	DESCRIPTION
63:48	RO	RSVD	Reserved.
47	RW	OF	Sticky counter overflow flag. Set to 1 when incrementing the CE count field results in unsigned integer overflow. 0: Counter has not overflowed. 1: Counter overflowed. The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and the counter overflow flag is set to 1. A direct write that modifies this bit or clears this bit to zero may indirectly set ERR0STATUS.OF to an UNKNOWN value.
46:32	RW	CEC	CE count. Incremented for each CE.
31:2	RO	RSVD	Reserved.
1	RW	RX_BUFF_ECC_ERR	RX buffer single bit ECC error observed.
0	RW	TX_BUFF_ECC_ERR	TX buffer single bit ECC error observed.

8.5.6.6 Error Record 0 Miscellaneous Register 1 (ERR0MISC1)

ERR0MISC1 is reserved.

Table 8-108: Error Record 0 Miscellaneous Register 1 (ERR0MISC1)

BIT	TYPE	FIELD	DESCRIPTION
63:0	RO	RSVD	Reserved.

8.5.6.7 Error Record 1 Feature Register (ERR1FR)

ERR1FR defines implemented features, and of the implemented features, which are software programmable.

ERR1FR is RO.

Table 8-109 summarizes ERR0FR; the Description column provides default field values.

Table 8-109: Error Record 1 Feature Register (ERR1FR) (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:20	RO	RSVD	Reserved.
19:18	RW	CEO	CE overwrite. Indicates the behavior when a second CE is detected after a first CE is recorded by the node. 00: Count CE if a counter is implemented. Keep the previous error syndrome. If the counter overflows, or no counter is implemented, ERR1STATUS.OF is set to 1.
17:16	RO	DUI	Error recovery interrupt for DEs. ERR0 does not support this feature.
15	RO	RP	Repeat counter. Indicates whether the node implements a repeat CE counter. 0: One CE counter is implemented.
14:12	RO	CEC	CE counter. Indicates whether the node implements a standard CE counter in ERR1MISC0. 100: The node implements a 16-bit CE counter in ERR1MISC0[47:32].
11:10	RO	CFI	Fault handling interrupt for CEs. Indicates whether the node implements a control for enabling fault handling interrupts on corrected errors. 10: The feature is controlled using ERR1CTLR.CFI.
9:8	RO	UE	In-band UE reporting. 0ERR1 does not support this feature.

Table 8-109: Error Record 1 Feature Register (ERR1FR) (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
7:6	RO	FI	Fault handling interrupt. Indicates whether the node implements a fault handling interrupt, and, if so, whether the node implements controls for enabling and disabling it. 10: The feature is controlled using ERR1CTLR.FI.
5:4	RO	UI	Error recovery interrupt for UEs. ERR1 does not support this feature.
3:2	RO	RSVD	Reserved.
1:0	RO	ED	Error reporting and logging. Indicates whether the node implements controls for enabling and disabling error reporting and logging. 10: The feature is controlled using ERR1CTLR.ED.

8.5.6.8 Error Record 1 Control Register (ERR1CTLR)

ERR1CTLR controls features such as error reporting and interrupt enables.

Table 8-110: Error Record 1 Control Register (ERR1CTLR) (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:11	RO	RSVD	Reserved.
10	R0	DUI	Error recovery interrupt for DEs. ERR0 does not support this feature.
9	RO	RSVD	Reserved.
8	RW	CFI	Fault handling interrupt for CEs enable. 0: Fault handling interrupt is not generated for CEs. 1: Fault handling interrupt is generated for CEs. When enabled: Because the node implements a CE counter, the fault handling interrupt is generated when the counter overflows and the counter overflow flag is set. See ERR0MISC0. The interrupt is generated even when the error syndrome is discarded because the error record recorded a higher priority error.
7:5	RO	RSVD	Reserved.
4	RO	UE	In-band UE reporting enable. ERR0 does not support this feature.

Table 8-110: Error Record 1 Control Register (ERR1CTLR) (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
3	RO	FI	Fault handling interrupt enable. 0: Fault handling interrupt is enabled. 1: Fault handling interrupt is not enabled. When enabled: Because the node implements a CE counter, the fault handling interrupt is generated when the counter overflows and the counter overflow flag is set. The interrupt is generated even when the error syndrome is discarded because the error record recorded a higher priority error.
2	RO	UE	UE recovery interrupt enabled. ERR1 does not support this feature.
1	RO	RSVD	Reserved.
0	RO	ED	Error reporting and logging enable. 0: Error reporting is disabled. 1: Error reporting is enabled. When disabled: The node behaves as if error detection is disabled, and no errors are recorded or signaled by the node.

8.5.6.9 Error Record 1 Primary Status Register (ERR1STATUS)

ERR1STATUS contains information about the error record.

Table 8-111: Error Record 1 Primary Status Register (ERR1STATUS) (Sheet 1 of 3)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. ERR1 does not support this feature.
30	RW	V	Status Register valid. 0: ERR1STATUS is not valid. 1: ERR1STATUS is valid. At least one error is recorded. This bit ignores writes if any of ERR1STATUS.{CE, DE, UE} are set to 1 and not cleared to 0 in the same write. This bit is read/write-one-to-clear. This bit resets to zero on a Cold reset.
29	RW	UE	UE. ERR1 does not support this feature.
28	RW	ER	Error Reported. ERR1 does not support this feature.

Table 8-111: Error Record 1 Primary Status Register (ERR1STATUS) (Sheet 2 of 3)

BIT	TYPE	FIELD	DESCRIPTION
27	RW	OF	Overflow. Indicates that multiple errors are detected. This bit is set to 1 when ERR1STATUS.UE == 0, ERR1STATUS.DE == 0, and the CE counter overflows. If this bit is nonzero, software must write 1 to this bit to clear this bit to zero when clearing ERR1STATUS.V to 0. This bit is not valid and reads UNKNOWN if ERR1STATUS.V is set to 0. This bit is read/write-one-to-clear.
26	RW	MV	Miscellaneous Registers Valid. 0: ERR1MISC0 and ERR0MISC1 are not valid. 1: The IMPLEMENTATION DEFINED contents of the ERR1MISC0 and ERR1MISC1 registers contain more information for an error recorded by this record. This bit ignores writes if any of ERR1STATUS.{CE, DE, UE} are set to 1 and the highest priority of these is not cleared to 0 in the same write. This bit is read/write-one-to-clear. This bit resets to zero on a Cold reset.
25:24	RW	CE	CE. 00: No errors are corrected. 01: Not supported. 10: At least one error is corrected. 11: Not supported. If this field is nonzero, then software must write ones to this field, to clear this field to zero, when clearing ERR1STATUS.V to 0. This field ignores writes if ERR1STATUS.OF is set to 1 and is not cleared to 0 in the same write. This field is not valid and reads UNKNOWN if ERR1STATUS.V is set to 0. This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an UNKNOWN value.
23	RW	DE	DE. ERR1 does not support this feature.
22	RW	PN	Poison. ERR0 does not support this feature.
21:20	RW	UET	UE Type. ERR0 does not support this feature.
19:16	RO	RSVD	Reserved.
15:8	RW	IERR	Implementation-defined error code. 0b0000 0000: No error. 0b0000 0001: iCRC or Data Link Layer Packet (DLLP) CRC mismatch.

Table 8-111: Error Record 1 Primary Status Register (ERR1STATUS) (Sheet 3 of 3)

BIT	TYPE	FIELD	DESCRIPTION
7:0	RW	SERR	Architecturally defined primary error code. Indicates the type of error. The primary error code can be used by a fault handling agent to identify an error without requiring a device-specific code. For example, to count CEs and handle threshold in software, or generate a short log entry. 0b0000 0000: No error. 0b0000 0001: IMPLEMENTATION DEFINED error.

8.5.6.10 Error Record 1 Address Register (ERR1ADDR)

ERR1ADDR is reserved.

Table 8-112: Error Record 1 Address Register (ERR1ADDR)

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.6.11 Error Record 1 Miscellaneous Register 0 (ERR1MISC0)

ERR1MISC0 contains a CE counter and other state information not available in the corresponding status and address error record registers.

Table 8-113: Error Record 1 Miscellaneous Register 0 (ERR1MISC0)

BIT	TYPE	FIELD	DESCRIPTION
63:48	RO	RSVD	Reserved.
47	RW	OF	Sticky counter overflow flag. Set to 1 when incrementing the CE count field results in unsigned integer overflow. 0: Counter has not overflowed. 1: Counter overflowed. The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and the counter overflow flag is set to 1. A direct write that modifies this bit or clears this bit to zero may indirectly set ERR1STATUS.OF to an UNKNOWN value.
46:32	RW	CEC	CE count. Incremented for each CE.
31:2	RO	RSVD	Reserved.
1	RW	DLLP_CRC_replay	DLLP CRC mismatch observed.
0	RW	iCRC_replay	iCRC mismatch observed.

8.5.6.12 Error Record 1 Miscellaneous Register 1 (ERR1MISC1)

ERR1MISC1 is reserved.

Table 8-114: Error Record 1 Miscellaneous Register 1 (ERR1MISC1)

BIT	TYPE	FIELD	DESCRIPTION
63:0	RO	RSVD	Reserved.

8.5.6.13 Error Record 2 Feature Register (ERR2FR)

ERR2FR defines implemented features, and of the implemented features, which are software programmable.

ERR2FR is RO.

Table 8-115 summarizes ERR0FR; the Description column provides default field values.

Table 8-115: Error Record 2 Feature Register (ERR2FR) (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:20	RO	RSVD	Reserved.
19:18	RW	CEO	CE overwrite. Indicates the behavior when a second CE is detected after a first CE is recorded by the node. 00: Count CE if a counter is implemented. Keep the previous error syndrome. If the counter overflows, or no counter is implemented, ERR2STATUS.OF is set to 1.
17:16	RO	DUI	Error recovery interrupt for DEs. ERR2 does not support this feature.
15	RO	RP	Repeat counter. Indicates whether the node implements a repeat CE counter. 0: One CE counter is implemented.
14:12	RO	CEC	CE counter. Indicates whether the node implements a standard CE counter in ERR2MISC0. 100: Implements a 16-bit CE counter in ERR2MISC0[47:32].
11:10	RO	CFI	Fault handling interrupt for CEs. Indicates whether the node implements a control for enabling fault handling interrupts on CEs. 10: The feature is controlled using ERR2CTLR.CFI.
9:8	RO	UE	In-band UE reporting. ERR2 does not support this feature.
7:6	RO	FI	Fault handling interrupt. Indicates whether the node implements a fault handling interrupt, and, if so, whether it implements controls for enabling and disabling it. 10: The feature is controlled using ERR2CTLR.FI.

Table 8-115: Error Record 2 Feature Register (ERR2FR) (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
5:4	RO	UI	Error recovery interrupt for UEs. ERR2 does not support this feature.
3:2	RO	RSVD	Reserved.
1:0	RO	ED	Error reporting and logging. Indicates whether the node implements controls for enabling and disabling error reporting and logging. 10: The feature is controlled using ERR2CTLR.ED.

8.5.6.14 Error Record 2 Control Register (ERR2CTLR)

ERR2CTLR controls features such as error reporting and interrupt enables.

Table 8-116: Error Record 2 Control Register (ERR2CTLR) (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31:11	RO	RSVD	Reserved.
10	R0	DUI	Error recovery interrupt for DEs. ERR2 does not support this feature.
9	RO	RSVD	Reserved.
8	RW	CFI	Fault handling interrupt for CEs enabled. 0: Fault handling interrupt is not generated for CEs. 1: Fault handling interrupt is generated for CEs. When enabled: Because the node implements a CE counter, the fault handling interrupt is generated when the counter overflows and the counter overflow flag is set. See ERR2MISC0. The interrupt is generated even when the error syndrome is discarded because the error record recorded a higher priority error.
7:5	RO	RSVD	Reserved.
4	RO	UE	In-band UE reporting enabled. ERR2 does not support this feature.
3	RO	FI	Fault handling interrupt enable. 0: Fault handling interrupt is enabled. 1: Fault handling interrupt is not enabled. When enabled: Because the node implements a CE counter, the fault handling interrupt is generated when the counter overflows and the counter overflow flag is set. The interrupt is generated even when the error syndrome is discarded because the error record recorded a higher priority error.

Table 8-116: Error Record 2 Control Register (ERR2CTLR) (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
2	RO	UI	UE recovery interrupt enabled. ERR0 does not support this feature.
1	RO	RSVD	Reserved.
0	RO	ED	Error reporting and logging enable. 0: Error reporting disabled. 1: Error reporting enabled. When disabled: The node behaves as if error detection is disabled, and the node does not record or signal errors.

8.5.6.15 Error Record 2 Primary Status Register (ERR2STATUS)

ERR2STATUS contains information about the error record.

Table 8-117: Error Record 2 Primary Status Register (ERR2STATUS) (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. ERR2 does not support this feature.
30	RW	V	Status Register valid. 0: ERR2STATUS is not valid. 1: ERR2STATUS is valid. At least one error is recorded. This bit ignores writes if any of ERR2STATUS.{CE, DE, UE} are set to 1 and not cleared to 0 in the same write. This bit is read/write-one-to-clear. This bit resets to zero on a Cold reset.
29	RW	UE	UE. ERR0 does not support this feature.
28	RW	ER	Error Reported. ERR0 does not support this feature.
27	RW	OF	Overflow. Indicates that multiple errors are detected. This bit is set to 1 when ERR2STATUS.UE == 0, ERR2STATUS.DE == 0, and the CE counter overflows. If this bit is nonzero, software must write 1 to this bit to clear this bit to zero when clearing ERR2STATUS.V to 0. This bit is not valid and reads UNKNOWN if ERR2STATUS.V is set to 0. This bit is read/write-one-to-clear.

Table 8-117: Error Record 2 Primary Status Register (ERR2STATUS) (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
26	RW	MV	Miscellaneous Registers Valid. 0: ERR2MISC0 and ERR2MISC1 are not valid. 1: The IMPLEMENTATION DEFINED contents of the ERR2MISC0 and ERR2MISC1 registers contain additional information for an error recorded by this record. This bit ignores writes if any of ERR2STATUS.{CE, DE, UE} are set to 1 and the highest priority of these is not cleared to 0 in the same write. This bit is read/write-one-to-clear. This bit resets to zero on a Cold reset.
25:24	RW	CE	CE. 00: No CE is recorded. 10: At least one CE is recorded. If this field is nonzero, then software must write ones to this field, to clear this field to zero, when clearing ERR0STATUS.V to 0. This field ignores writes if ERR2STATUS.OF is set to 1 and is not cleared to 0 in the same write. This field is not valid and reads UNKNOWN if ERR0STATUS.V is set to 0. This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an UNKNOWN value.
23	RW	DE	DE. ERR2 does not support this feature.
22	RW	PN	Poison. ERR2 does not support this feature.
21:20	RW	UET	UE Type. ERR2 does not support this feature.
19:16	RO	RSVD	Reserved.
15:8	RW	IERR	Implementation-defined error code. 0b0000 0000: No error. 0b0000 0001: Link error requires retraining.
7:0	RW	SERR	Architecturally defined primary error code. Indicates the type of error. The primary error code may be used by a fault handling agent to identify an error without requiring a device-specific code, for example, to count and threshold CEs in software, or generate a short log entry. 0b0000 0000: No error. 0b0000 0001: IMPLEMENTATION DEFINED error.

8.5.6.16 Error Record 2 Address Register (ERR2ADDR)

ERR2ADDR is reserved.

Table 8-118: Error Record 2 Address Register (ERR2ADDR)

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.6.17 Error Record 2 Miscellaneous Register 0 (ERR2MISC0)

ERR2MISC0 contains a CE counter and other state information not available in the corresponding status and address error record registers.

Table 8-119: Error Record 2 Miscellaneous Register 0 (ERR2MISC0)

BIT	TYPE	FIELD	DESCRIPTION
63:48	RO	RSVD	Reserved.
47	RW	OF	Sticky counter overflow flag. Set to 1 when incrementing the CE count field results in unsigned integer overflow. 0: Counter has not overflowed. 1: Counter overflowed. The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and the counter overflow flag is set to 1. A direct write that modifies this bit or clears this bit to zero may indirectly set ERR2STATUS.OF to an UNKNOWN value.
46:32	RW	CEC	CE count. Incremented for each CE.
31:3	RO	RSVD	Reserved.
2	RW	B2B_NAK_threshold	Back-to-back DLLP NAK threshold exceeded.
1	RW	CRC_error_threshold	RX buffer single-bit ECC error observed.
0	RW	link_framing_error	Link framing error observed.

8.5.6.18 Error Record 2 Miscellaneous Register 1 (ERR2MISC1)

ERR2MISC1 is reserved.

Table 8-120: Error Record 2 Miscellaneous Register 1 (ERR2MISC1)

BIT	TYPE	FIELD	DESCRIPTION
63:0	RO	RSVD	Reserved.

8.5.6.19 Error Record 3 Feature Register (ERR3FR)

ERR3FR defines implemented features, and of the implemented features, which are software programmable.

ERR3FR is RO.

Table 8-121 summarizes ERR3FR; the Description column provides default field values.

Table 8-121: Error Record 3 Feature Register (ERR3FR)

BIT	TYPE	FIELD	DESCRIPTION
31:20	RO	RSVD	Reserved.
19:18	RW	CEO	CE overwrite. ERR3 does not support this feature.
17:16	RO	DUI	Error recovery interrupt for DEs. ERR0 does not support this feature.
15	RO	RP	Repeat counter. ERR3 does not support this feature.
14:12	RO	CEC	CE counter. ERR3 does not support this feature.
11:10	RO	CFI	Fault handling interrupt for CEs. ERR3 does not support this feature.
9:8	RO	UE	In-band UE reporting. ERR3 does not support this feature.
7:6	RO	FI	Fault handling interrupt. ERR3 does not support this feature.
5:4	RO	UI	Error recovery interrupt for UEs. Indicates whether the node implements an error recovery interrupt, and, if so, whether it implements controls for enabling and disabling it. 10: The feature is controlled using ERR3CTLR.UI.
3:2	RO	RSVD	Reserved.
1:0	RO	ED	Error reporting and logging. Indicates whether the node implements controls for enabling and disabling error reporting and logging. 10: The feature is controlled using ERR3CTLR.ED.

8.5.6.20 Error Record 3 Control Register (ERR3CTLR)

ERR3CTLR controls features such as error reporting and interrupt enables.

Table 8-122: Error Record 3 Control Register (ERR3CTLR)

BIT	TYPE	FIELD	DESCRIPTION
31:11	RO	RSVD	Reserved.
10	R0	DUI	Error recovery interrupt for DEs. ERR0 does not support this feature.
9	RO	RSVD	Reserved.
8	RW	CFI	Fault handling interrupt for CEs enable. ERR3 does not support this feature.
7:5	RO	RSVD	Reserved.
4	RO	UE	In-band UE reporting enabled. ERR3 does not support this feature.
3	RO	FI	Fault handling interrupt enable. ERR3 does not support this feature.
2	RO	UI	UE recovery interrupt enable. 0: Error recovery interrupt disabled. 1: Error recovery interrupt enabled. When enabled: The error recovery interrupt is generated for all detected UEs that are not deferred. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.
1	RO	RSVD	Reserved.
0	RO	ED	Error reporting and logging enable. 0: Error reporting disabled. 1: Error reporting enabled. When disabled: The node behaves as if error detection is disabled, the node does not record or signal errors.

8.5.6.21 Error Record 3 Primary Status Register (ERR3STATUS)

ERR3STATUS contains information about the error record.

Table 8-123: Error Record 3 Primary Status Register (ERR3STATUS) (Sheet 1 of 2)

BIT	TYPE	FIELD	DESCRIPTION
31	RW	AV	Address Valid. ERR3 does not support this feature.
30	RW	V	Status Register valid. 0: ERR3STATUS is not valid. 1: ERR3STATUS is valid. At least one error is recorded. This bit ignores writes if any of ERR0STATUS.{CE, DE, UE} are set to 1, and is not being cleared to 0 in the same write. This bit is read/write-one-to-clear. This bit resets to zero on a Cold reset.
29	RW	UE	UE. 0: No errors are detected, or all detected errors are either corrected or deferred. 1: At least one detected error was not corrected and not deferred. If this bit is nonzero, software must write 1 to this bit to clear this bit to zero when clearing ERR3STATUS.V to 0. This bit ignores writes if ERR3STATUS.OF is set to 1 and is not being cleared to 0 in the same write. This bit is not valid and reads UNKNOWN if ERR3STATUS.V is set to 0. This bit is read/write-one-to-clear.
28	RW	ER	Error Reported. ERR0 does not support this feature.
27	RW	OF	Overflow. Indicates that multiple errors are detected. This bit is set to 1 when ERR3STATUS.UE == 0, ERR3STATUS.DE == 0, and the CE counter overflows. If this bit is nonzero, software must write 1 to this bit to clear this bit to zero when clearing ERR3STATUS.V to 0. This bit is not valid and reads UNKNOWN if ERR3STATUS.V is set to 0. This bit is read/write-one-to-clear.
26	RW	MV	Miscellaneous Registers Valid. ERR3 does not support this feature.
25:24	RW	CE	CE. ERR3 does not support this feature.
23	RW	DE	DE. ERR3 does not support this feature.
22	RW	PN	Poison. ERR3 does not support this feature.

Table 8-123: Error Record 3 Primary Status Register (ERR3STATUS) (Sheet 2 of 2)

BIT	TYPE	FIELD	DESCRIPTION
21:20	RW	UET	UE Type. Describes the component state after detecting or consuming a UE. 01: UE, Unrecoverable Error (UEU). If this field is nonzero, software must write ones to this field to clear this field to zero when any of: ERR3STATUS.V is cleared to 0. ERR3STATUS.UE is cleared to 0. This field ignores writes if any of ERR3STATUS.{CE, DE, UE} are set to 1, and the highest priority of these is not cleared to 0 in the same write. This field is not valid and reads UNKNOWN when any of: ERR3STATUS.V is set to 0. ERR3STATUS.UE is set to 0. This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an UNKNOWN value.
19:16	RO	RSVD	Reserved.
15:8	RW	IERR	Implementation-defined error code. 0b0000 0000: No error. 0b0000 0001: TX buffer double-bit ECC error. 0b0000 0010: RX buffer double-bit ECC error. 0b0000 0011: RX buffer overrun. 0b0000 0100: Link retraining failure. 0b0000 0101: Link Retraining Threshold. 0b0000 0110: Link Rate Degradation Failure. 0b0000 0111: SeqNum Color Overflow.
7:0	RW	SERR	Architecturally defined primary error code. Indicates the type of error. The primary error code can be used by a fault handling agent to identify an error without requiring a device-specific code, for example, to count CEs and handle thresholds in software, or to generate a short log entry. 0b0000 0000: No error. 0b0000 0001: IMPLEMENTATION DEFINED error.

8.5.6.22 Error Record 3 Address Register (ERR3ADDR)

ERR3ADDR is reserved.

Table 8-124: Error Record 3 Address Register (ERR3ADDR)

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.6.23 Error Record 3 Miscellaneous Register 0 (ERR3MISC0)

ERR3MISC0 is reserved.

Table 8-125: Error Record 3 Miscellaneous Register 0 (ERR3MISC0)

BIT	TYPE	FIELD	DESCRIPTION
31:0	RO	RSVD	Reserved.

8.5.6.24 Error Record 3 Miscellaneous Register 1

ERR3MISC1 is reserved.

Table 8-126: Error Record 3 Miscellaneous Register 1 (ERR3MISC1)

BIT	TYPE	FIELD	DESCRIPTION
63:0	RO	RSVD	Reserved.

8.5.6.25 Error Group Status Register (ERRGSR)

This register shows the status of the ALI error record status registers. When an interrupt occurs, software can check this register to tell which error record or records caused the interrupt.

ERRGRSR.{Errs3, Errs2, Errs1, Errs0} each contain a copy of the V bit of the corresponding error record status register.

ERRGSR is RO.

Table 8-127: Error Group Status Register (ERRGSR)

BIT	TYPE	FIELD	DESCRIPTION
31:4	RO	RSVD	Reserved.
3	RO	Errs3	Error Record 3 status; a read-only copy of ERR3STATUS.V. 0: No error. 1: One or more errors.
2	RO	Errs2	Error Record 2 status; a read-only copy of ERR2STATUS.V. 0: No error. 1: One or more errors.
1	RO	Errs1	Error Record 1 status; a read-only copy of ERR1STATUS.V. 0: No error. 1: One or more errors.
0	RO	Errs0	Error Record 0 status; a read-only copy of ERR0STATUS.V. 0: No error. 1: One or more errors.

Clocks and Reset

9

This chapter describes clocks in the Altra Max processor, along with Altra Max processor resets. The chapter covers these topics:

9.1 Overview

An Altra Max processor provides several clocks in four main clock domains.

For information about the relationships among system clocks, their frequencies, and power management, see *Chapter 11, "Power Management."*

An Altra Max processor provides a Power On Reset (POR).

9.2 Clocks

Each clock domain has one or more separate Phase-Locked Loops (PLLs):

- CMI clock domain.

  This domain has one PLL in the PCP power domain. The CMI clock generates all PCP-related clocks except the cores.
- Core clock domain.

  Each group of four cores, called a Quad CPU (QCPU), has a PLL in the PCP power domain; 32 QCPU PLLs provide individual clocks to the 128 Altra Max cores. These core clocks are programmable.
- DDR clock domain.

  Each of the eight MCUs has a dedicated PLL in the SoC power domain. The DDR clock is programmable.
- SoC clock domain.

  This domain has a PLL in the SoC power domain that generates SoC-related clocks, such as the AHBC clock, SMpro clock, and so on, except for those in the DDR clock domain.

  Three additional PLLs in the SoC clock domain generate clocks for other internal buses, processing elements, and low-frequency peripheral interfaces.

Depending upon the required clock frequencies, some blocks locally divide down the clocks generated by the PLLs. The dividers for the main clocks are grouped with the PLLs. Other dividers used to divide down a main clocks are local to the blocks.

These clocks are generated in the Altra Max processor:

- PCP_PLL clocks.
- QCPUn_PLL (n is from 0 to 19).
- MCUn_PLL (n is from 0 to 7).
- SoC PLL clocks:
  - PCIe aggregating fabric clock (RcA: 1.5625 GHz).
  - CCIX-AXI clock (781.25 MHz).
  - PCIe-AXI clock (500 MHz).
  - SYS-AXI clock (400 MHz).
  - SMpro clock (400 MHz).
  - PMpro clock (400 MHz).
  - AHBC clock (200 MHz).

- DAP clock (400 MHz) on SYS.
- Generic counter/timer clock (50 MHz).

9.2.1 PCP_PLL Clocks

The PCP_PLL clocks control the CMI clock.

- For the CMI clock, DVFS is performed through the PLL output.
- CMI clock frequency (GCLK) ranges from 1.0 GHz to 2.5 GHz. The nominal CMI clock frequency is 2.0 GHz. The turbo CMI clock frequency is up to 2.5 GHz, and the minimum CMI clock frequency is 1 GHz. The frequency step size is 50 MHz.
- The CMI clock can be gated (when gated, the CMI is in the deep sleep state).
- The CMI clock is used for the CMI and GICs.
- Reduced frequency clocks in the PCP, such as the debug clock, can use a fixed divided-down clock from the CMI clock. Dividers are local to these blocks.

PMpro controls the PCP_PLL.

9.2.2 QCPU_PLL Clocks

Each group of four cores is clocked using one of the 32 QCPU_PLL clocks.

- All QCPU_PLL clocks run at the same frequency, which changes dynamically to meet power and thermal requirements.
- Each CPUn clock (CPU core clock) is programmable to use clock skipping to step down the clock from 32/32 to 9/32 of the QCPU_PLL clock frequency. Each core has its own local DFS capability.
- The nominal core clock is 2.6 GHz. The turbo frequency is up to 3.0 GHz, and the minimum core clock frequency is 1 GHz. The frequency step size is 50 MHz. See *Section 11.5, "Performance Feedback Counters,"* for a description of turbo mode.
- The QCPU_PLL clocks run at the highest frequency requested across the CPU cores.
- The CPUn clocks are programmable to be gated off.

The QCPU_PLL clocks are controlled by PMpro registers. PMpro must be brought out of hardware reset before the cores.

9.2.3 MCUn_PLL Clocks

Each DDR interface is clocked by an MCUn_PLL clock.

- Divided-by-two phase-aligned MCUn clock (clk in the MCUs). The supported DDR4 rates are 1600 MT/s, 1866 MT/s, 2133 MT/s, 2400 MT/s, 2667 MT/s and 3200 MT/s (1600 MHz).
- MCUndiv2 clock (clkdiv2 in the MCUs):
  - A version of MCUn clock.
  - The clock is gated while in self-refresh (S3 state: deep sleep state)
- MCUn_PLL is located in the MCU.
- The PLL control, clock, and reset control registers are in a separate block. Registers are accessed through the CB.
- Each MCU has level shifters and isolation logic for inputs from the CMI/PCP power domain, including PLL control signals.

9.2.4 SoC PLL Clocks

SMpro autonomously controls the SOC_PLL, which provides these clocks:

- PCIe RcA Aggregating Fabric clock (1.5625 GHz).
- PCIe-AXI Clock (500 MHz).

9.3 Reset

The Altra Max processor provides a POR that resets all SoC components and devices.

Refer to *Altra Max Datasheet*, which contains detailed reset sequences for 1P and 2P systems.

Boot Process

10

This chapter describes the Altra Max boot process. The chapter covers these topics:

10.1 Overview

This chapter describes the Altra Max Secure Boot Solution, which covers Root of Trust (RoT) from power-on to booting the operating system. The solution supports a TPM, but no TPM is necessary. This boot solution complies with existing standards, such as NIST SP 800-147B, *BIOS Protection Guidelines for Servers*; *Arm Trusted Board Boot Requirement (TBBR) Specification* (ARM DEN0006C-1), UEFI secure boot (UEFI Version 2.3.1 Errata), and Microsoft secure boot requirements.

SMpro handles overall system management. Initially booting the system is one of the primary SMpro responsibilities. The Altra Max processor always boots from an untampered ROM called the boot ROM. The Altra Max processor requires an external EEPROM to be attached to the SMpro I^2C interface from which the processor loads its 256-bit bootstrap vector. In Secure boot mode, the bootstrap is loaded from eFuses.

10.2 Boot Process Features

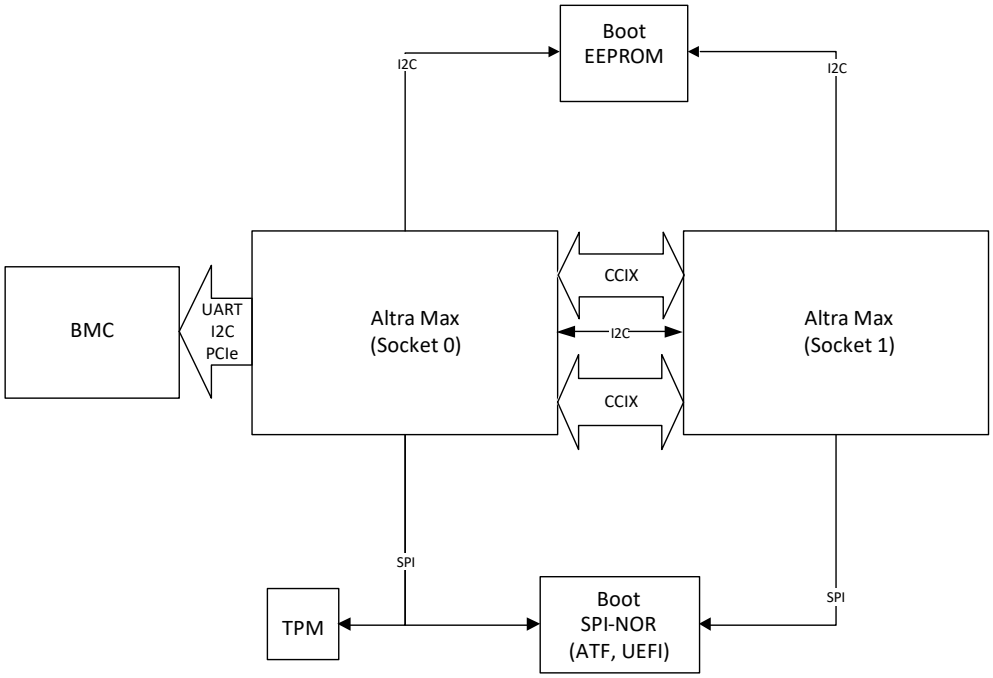
The Altra Max processor provides numerous enhanced security features:

- A TMM eFuse array to enable various security features.
- Anti-roll back counter support stored in the TMM eFuse array to blacklist firmware with known security vulnerabilities.
- Debug authentication, for example, the ability to sign firmware with fine-grained debug control masks for generating debug certificates intended to re-enable specific hardware debug features in production fused parts.
- HUK, a unique device ID/key in eFuse that enables binding firmware to a specific part, for example, signing engineering and debug builds that boot only on specific platforms.
- A key revocation mechanism for periodically rotating root public keys in fuses.
- TPM support:
  - TPM is supported over SPI and complies with the TCG TPM Interface specification.
  - Altra Max implements Static RoT Measurement (SRTM), with the first measurements logged from early authenticated boot firmware to ensure that logging is secure and reliable.
  - Along with logging hashes of firmware images at each boot stage, the firmware measures and logs hashes of unsigned configuration data: some eFuse arrays; UEFI platform configuration data; UEFI variable storage; and so on.
  - Optional: Isolated TPM service in the secure world is exposed using a firmware interface compliant with the TCG Command Response Buffer (CRB) protocol.
- 2P authentication:
  - Leverages socket and configuration information for socket authentication (for example, correct eFuse configuration, debug settings, and SKU).
  - Everything is measured and logged to TPM.

10.3 Boot Peripherals

Figure 10-1 shows the boot peripherals in a 2P system. (In a 1P system, the processor labeled Socket 1 and the CCIX and I^2C interfaces connecting the two processors are absent.) The diagram does not show DIMM Serial Presence Detect (SPD).

Figure 10-1: Boot Peripherals (2P System)



- Boot EEPROM.
  - Contains SMpro and PMpro firmware
  - Accessed by both sockets in 2P systems; I^2C arbitrates access.
- BMC host interfaces (UART, I^2C, PCIe); connects only to the primary socket.
- Out-of-band I^2C is used to communicate between sockets before 2P CCIX link initialization.
- TPM is owned by and accessible only by the primary socket (Socket 0).
- Boot SPI-NOR flash, which contains ATF, UEFI, and variable storage. See *Section 10.16.2, "SPI-NOR Storage,"* for information about SPI-NOR storage.
- DIMM SPD: Each socket can access only its local SPD.

10.4 High-Level Boot Flow

This section describes the design of the secure boot flow for a 2P configuration. Depending upon the system designer, the actual implementation can differ slightly. For a 1P configuration, interactions with the scondary socket are ignored.

The Altra Max secure boot flow implements the Arm TBBR specification, which describes how to maintain a trustworthy boot process. The boot flow establishes a Chain of Trust (CoT) using Public-Key-Cryptography Standards (PKCS). All firmware images up to and including the normal world bootloader must be authenticated.

If a tampered binary is detected, the CoT breaks and the boot fails. *Figure 10-2* shows the boot flow overview without authentication. Subsequent sections describe authentication in more detail and provide detailed boot flows.

Figure 10-2 is simplified to illustrate the high-level flow, so the figure does not show various details, such as detailed certificate and image authentication flows.

10.4.1 SMpro Firmware

The Altra Max secure boot solution covers various boot phases that involve SMpro, PMpro, and the Armv8 processor (running ATF, UEFI, OS loader, and the underlying OS.

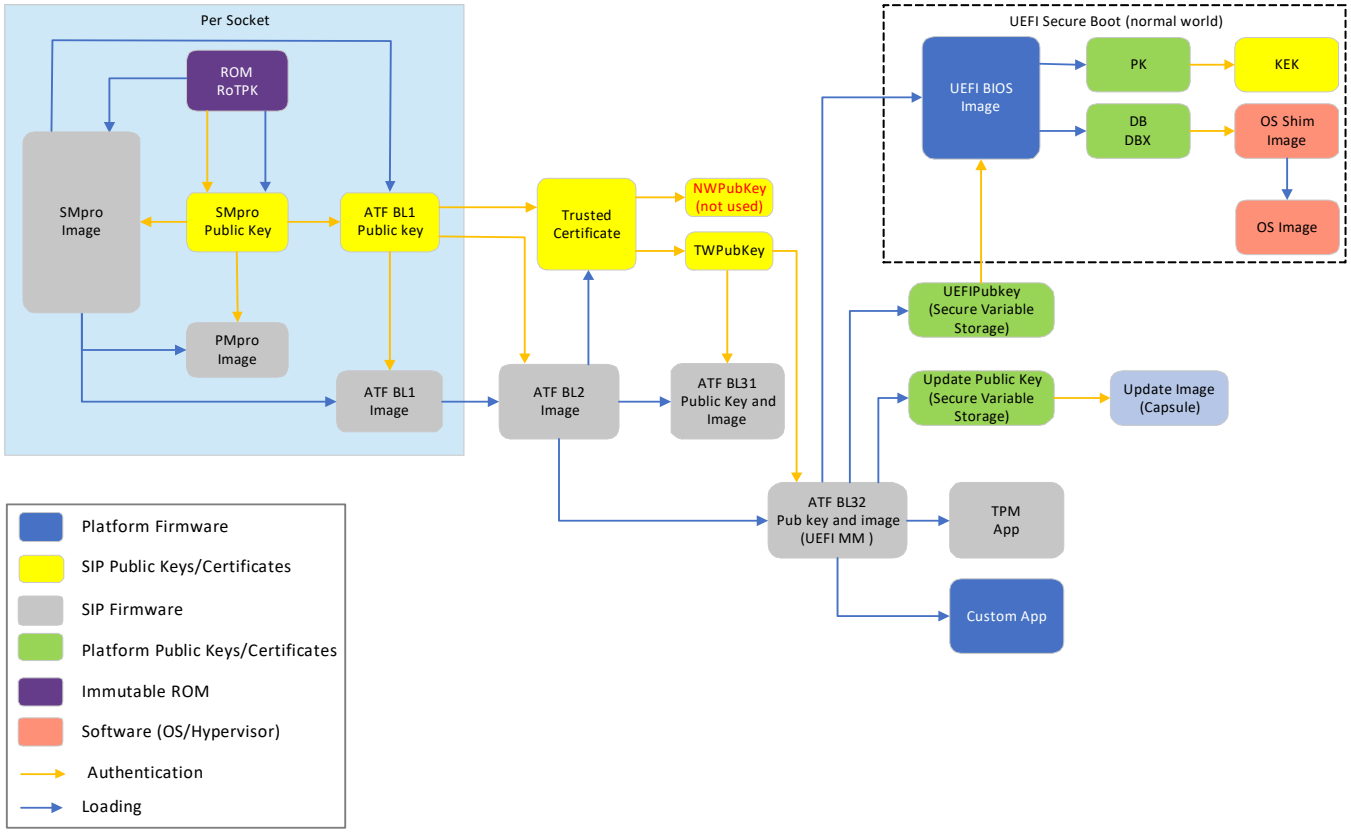
The SMpro firmware comprises these components:

- SMpro certificates:
  - Key certificate.
  - Content certificate.
  - Special boot certificate.
- SMpro firmware image:
  - Contains code for blowing fuses and certificate authentication.
  - Loads or simply branches to the next SMpro phase.
  - Contains all other boot code (TPM, 2P authentication, CMI initialization, PMpro initialization, CPM initialization, and so on).
  - Contains run-time firmware, such as RAS, BMC interface, and so on.
- SMpro revocation image (only present as part of key revocation flow):
  - This contains code for blowing fuses and certificate authentication.
  - This image will locate and authenticate the replacement SMpro SLIM and associated certificates.
  - Overwrites and rearranges the EEPROM before rebooting, to proceed with a normal boot on the next boot.
  - This image cannot boot past SMpro.

10.4.2 Simplified High-Level Dual-Socket Platform (2P) Boot Flow Illustration

Figure 10-2 illustrates a high level (simplified) normal secure boot flow, starting from ROM to booting the OS.

Figure 10-2: Simplified High-Level 2P Boot Flow



The blue box at the upper left describes a high-level view of the authentication flow that occurs on each socket in parallel.

- SMpro firmware runs on the SMpro processor of each socket.
- PMpro firmware runs on the PMpro processor of each socket.
- ATF BL1 Firmware runs on core 0 of each socket.

The flow beyond this left blue highlight box represents the SMP authentication flow that runs only on the primary socket.

Each authentication stage is also a point at which image hashes are measured and logged to TPM.

This flow implements numerous sync points for handshaking/coordination between the two sockets. A 2P authentication flow occurs during the SMpro phase to authenticate the 2P link. Measurements of signed code and unsigned data are logged to the TPM.

Note: The initial Altra Max boot flow does not authenticate secure applications such as UEFI MM and only authenticates UEFI. More comprehensive authentication will be added for secure application and UEFI boot stages.

10.5 SMpro Boot Responsibilities and Tasks

The SMpro microcontroller is the boot processor for the Altra Max processor. In 2P systems, the SMpro in each processor boots its associated Altra Max processor.

SMpro is the RoT for the system, and so serves as the foundation of system security. For more information about the role of SMpro in system security, see *Chapter 7, "Security."*

The SMpro firmware is divided into two images:

- SMpro SEC image:
  - Contains code for blowing fuses and certificate authentication.
  - Contains certificates for the boot ROM.
  - Loads or branches to the next SMpro phase.
- SMpro firmware image:
  - Contains all other boot code (TPM, 2P authentication, CMI initialization, PMpro initialization, CPU initialization, and so on).
  - Contains run-time firmware for RAS, BMC interface, and so on.

This split extends complex security tasks typically handled by the ROM into an authenticated firmware phase for flexibility. Future generations of the processor may move these complex security tasks into ROM after the tasks are verified.

Section 10.14.3, "Debug-Disable Extension Fields," describes the debug-disable extension fields.

10.6 Boot ROM Code Responsibilities

The boot ROM code reads the manufacturing eFuse array to determine whether TMM is enabled and to determine the current eFuse LCS. If TMM is enabled, the boot ROM code must also query the TMM eFuse to ensure that the Altra Max processor is not in the Returned Material Authorization (RMA) LCS. The RMA LCS is beyond the scope of this manual.

If TMM is enabled and the system is not in the RMA LCS, the boot ROM performs these actions.

- Programs the OTP CSR space based on eFuses:
  - HUK.
  - Root Of Trusted Public Key (RoTPK).
  - Integrity Check Value (ICV) counter.
  - Everything else populated after ROM.
- Locks the TMM eFuses unless a valid certificate for blowing eFuses is present (see *Section 10.8.2, "SMpro Special Boot Modes"*).
- Sets and locks the debug control register unless a valid debug certificate is present (see *Section 10.8.2, "SMpro Special Boot Modes"*).

10.7 Boot ROM Process Flow

After the Altra Max processor Bootstrap Controller (BSC) brings the SMpro microcontroller out of reset, the BSC begins fetching and executing instructions from the on-chip boot ROM. The cryptographic hardware is held in reset until the OTP CSR space is populated. These steps summarize the boot ROM process flow:

1. The boot ROM reads the manufacturing eFuse array to determine whether TMM is enabled and whether the Altra Max processor is in the RMA LCS.
2. If TMM is enabled, the boot ROM queries the TMM eFuse to populate the OTP CSR space based on eFuse. HUK, RoTPK, ICV Counter, DCU mask, and so on.
3. Initialize the cryptographic hardware.
4. Read eFuse[ROTPK_INDEX] to determine which RoTPK to use. If eFuse[ROTPK_INDEX]=0, use the RoTPK embedded in ROM. Otherwise, use the appropriate RoTPK in eFuse specified by the eFuse[ROTPK_INDEX] value.
5. Authenticate the SMpro Key Certificate using the cryptographic hardware with the appropriate Hashed RoTPK.
6. If the SMpro Key Certificate is genuine, authenticate the SMpro Content Certificate using the cryptographic hardware with the Hashed SMpro Public Key in the SMpro Key Certificate content.
7. Locate and load the SMpro image from EEPROM into the DRAM block by block to calculate its SHA256 hash value.
8. Compare the calculated hash value with the hashed value in the SMpro Content Certificate to authenticate SMpro image.
9. If the comparison mismatches, SMpro authentication fails, asserting GPIO_FAULT and going into an infinite loop to stall the system.
10. If the comparison matches, the SMpro image is authentic; locate and load only the SMpro SEC phase to Instruction RAM (IRAM).
11. Anti-rollback checking: compare anti-rollback version in SMpro image with the ICV counter in eFuse; if the anti-rollback version is greater than the ICV counter, set the WRITE_TMM_EFUSE flag.
12. Re-enable debug checking:
    a) Read the extension field of SMpro Content Certificate to determine whether this is a re-enable debug certificate.
    b) If it is not, read the DCU mask from eFuse and inverted write to TMMDBGCR.
    c) If it is, go to Step 13. TMMDBGCR is written in the SMpro SEC phase.
13. Debug disable certificate checking:
    a) Read the extension field of the SMpro Content Certificate to determine whether this is a revocation of the Debug disable certificate.
    b) If it is, set WRITE_TMM_EFUSE flag and go to Step c).
    c) If it is not, check if WRITE_TMM_EFUSE is set, go to Step 14. Otherwise, lock down TMM eFuses by setting TMMLOCKR.
14. SMpro jumps to IRAM and starts running the SEC phase from IRAM.

10.8 Boot Tasks

During boot, the SMpro microcontroller performs these tasks:

- Runs ROM code, which loads and authenticates the SMpro firmware and the key certificate and content certificate.
- Loads, authenticates, and initializes the PMpro microcontroller.

- Discovers populated DIMM slots using SPD.
- Loads and authenticates the next boot phase to maintain a trusted certificate chain.
- Initializes local socket power.
- Initializes the local CPMs and CMI and bus configurations.
- Configures the overall system address map and system coherency
- For 2P systems, initializes and authenticates 2P links (dedicated CCIX links).
- Manages 2P boot flow synchronization, such as I^2C EEPROM and SPI NOR arbitration and mux control.

10.8.1 SMpro Secure Boot Flow

When the BMC/platform applies power, the immutable ROM starts to run on the primary System Control Processor (SCP), called SMpro. The Altra Max SCP comprises two microcontrollers, one for system management (SMpro) and the other for power management (PMpro). SMpro is considered the "boot processor" because it is first to come out of reset, and it initializes PMpro and the CPU cores in the PCP.

The SMpro ROM firmware, which accesses only the EEPROM, performs these operations in the "ROM boot phase":

1. Loads the SLIM header key certificate and content certificate.
   a) Authenticates the SLIM header key hash using the fused RoTPK hash.
   b) Authenticates the content certificate using the SLIM header key hash.
2. Loads the SLIM header, SLIM file table, and certificates in the SLIM image.
   a) Authenticates these contents using the SLIM header key hash.
3. The SMpro key and content certificate (loaded during Step 2) authenticates the SMpro public key hash using the fused RoTPK hash.
4. Loads the SMpro firmware image based on authenticated SLIM file table.
   a) Authenticates content certificate and SMpro image using SMpro public key hash.

Note: The certificates and images are all loaded from EEPROM into SMpro RAM

If the SMpro firmware authenticates successfully, the SMpro ROM firmware branches and runs the SMpro firmware. Otherwise, the boot process is halted with no other communication except a blinking LED and an error message printed to the console. If there is no POWER_OK acknowledgment from the SMpro to the BMC, the BMC must assume an authentication failure and log an event.

The SMpro firmware performs these operations to boot PMpro:

1. Loads the PMpro key and content certificate:
   a) Authenticates PMpro key hash using SMpro public key hash.
2. Loads the PMpro firmware image based on authenticated SLIM file table:
   a) Authenticates content certificate and image using the PMpro public key hash.
3. Brings PMpro processor out of reset; PMpro begins running the PMpro firmware.

Note: These certificates and images are loaded from EEPROM into the PMpro RAM.

- Next, the SMpro boots the ATF, which is stored in SPI-NOR boot storage. See *Section 10.16.2, "SPI-NOR Storage,"* for more information about SPI-NOR storage, including its layout. The ATF performs these operations:

1. Loads the ATF BL1 key and content certificate.
   a) Authenticates ATF BL1 key hash using SMpro public key hash
2. Loads the ATF BL1 Firmware image.
   a) Authenticates content certificate and image using ATF BL1 public key hash.
3. Brings core 0 out of reset, which begins running ATF BL1 Firmware.

Note: These certificates and images are loaded from SPI-NOR into OCM.

If any preceding step fails authentication, the SMpro firmware halts the boot process and updates the SMpro I^2C secondary device interface to provide boot failure status to BMC.

10.8.2 SMpro Special Boot Modes

SMpro follows different boot flows when certain certificates are present.

10.8.2.1 Normal Boot Mode with Debug-Disable Mode

The debug-disable scenario requires special considerations because the debug disable certificate is flashed independently of the SMpro SLIM image. In this scenario, a customer can disable fuses.

Customers typically receive platforms without flashed debug disable certificates. A customer wanting to flash the debug disable certificates must work with Ampere Computing as follows:

- Ampere Computing provides a debug disable certificate for the Altra Max processor (multiple certificates having unique debug disable masks may be provided).
- Ampere Computing provides a special firmware update flow to inject this image into the top of the EEPROM and reboot the system. This can be both in-band and Out of Band (OOB) update mechanisms. For in-band updates, the debug disable certificates are likely to be delivered as a UEFI update capsule.

If a debug disable certificate is present at boot, ROM keeps fuses unlocked, unless the debug fuses are already blown. The SMpro firmware must delete the debug-disable certificates immediately after blowing fuses and then reboot. To avoid unintended debug fuse burning, for example, as a result of inadvertent or malicious flashing of debug-disable certificates, the SPECIAL_BOOT pin must be asserted to enter this boot mode.

For this scenario, the boot ROM responsibilities include:

- Ensuring that a debug-disable certificate is present at the fixed location at the top of the EEPROM.
- Ensuring that the SPECIAL_BOOT pin is asserted.
- Authenticating the SMpro firmware (smkeycert, smcontcert, and image).
- If the debug-disable certificate is valid, keeping the fuses unlocked to be blown by the SMpro firmware.
- Otherwise, locking the fuses and continuing to boot.
- Locking the secure debug settings as specified by the current TMM eFuse debug control mask.

See the high-level boot flow diagram in *Section 10.4, "High-Level Boot Flow."*

10.8.2.2 Debug (Enable) Re-Enable Mode

This customer use case is to flash an engineering debug SLIM build generated to unlock debugging on a firmware release or branch. Customers can flash this build, provided by Ampere Computing, using in-band or OOB mechanisms.

For this scenario, the boot ROM responsibilities include:

- Ensuring that a debug enable certificate is present in the authenticated SLIM file table.
- Ensuring that the SPECIAL_BOOT pin is asserted.
- Authenticating the SMpro firmware (smkeycert, smcontcert, and image).
- If the debug enable certificate is valid (tied to SMpro):
  - If the debug enable certificate is tied to a non-zero HUK, and if the HUK is correct, keep debug settings unlocked (to be set by the SMpro firmware).
  - Else fail to boot.
- Else
  - If the SPECIAL_BOOT pin is asserted, keep debug settings unlocked (to be set by the SMpro firmware).
  - Else fail to boot.
- Else fail to boot.
- Lock fuses.

See the high-level boot flow diagram in *Section 10.4, "High-Level Boot Flow."*

10.8.3 Post-Boot Tasks

After performing its initial boot tasks, SMpro performs these tasks:

- Provides RAS services for firmware-first handling hardware errors, threshold counting, and error injection.
- Manages the out-of-band interface to the BMC.

10.9 PMpro Boot Responsibilities

PMpro firmware runs on a dedicated PMpro microcontroller per socket.

Run-time:

- Handles local socket power/performance management using ACPI, for example, Collaborative Processor Performance Controls (CPPC).
- Local power/thermal throttling using a BMC I^2C interface.

10.10 Arm Trusted Firmware (ATF) Secure Boot

On the primary socket logical core 0, ATF BL1 runs from OCM as authenticated in the preceding boot stage. Note that this differs from the TBBR specification, which assumes that BL1 runs from a secure ROM and does not require authentication. Another difference is that the RoTPK maps to the ATF BL1 public key, which the SMpro firmware loads and authenticates from SPI-NOR. The authenticated ATF BL1 public key hash is stored in OCM to be referenced later by the ATF BL1 Firmware for subsequent authentication phases.

ATF BL1 handles DDR initialization on each socket. After DDR initialization finishes and appropriate sync points are reached, only the primary socket proceeds, while secondary socket logical core 0 enters the low-power idle state (that is, WFI) waiting on the Power State Coordination Interface (PSCI) CPU ON call from the OS software before running code.

The ATF BL1 Firmware (primary socket only) performs these operations:

1. Fetches the ATF BL1 public key hash, which SMpro passes into a known location in OCM.
2. Fetches the ATF BL2 content certificate and authenticates it using the ATF BL1 public key hash.
3. Fetches the BL2 image and authenticates the BL2 image and content certificate using the ATF BL1 public key hash.

After successfully verifying ATF BL2, the primary socket logical core 0 runs ATF BL2. If any preceding step fails authentication, ATF BL1 Firmware halts the boot process and communicate with the SMpro firmware. The BMC can query the SMpro for any boot failure condition and log an appropriate event.

The ATF BL2 Firmware performs these operations:

1. Loads the ATF "trusted certificate" from SPI-NOR.
2. Authenticates the ATF "trusted certificate" using the ATF BL1 public key hash.
3. Fetches the trusted world public key (TWPubKey) from the authenticated ATF "trusted certificate."
4. Fetches the BL31 key certificate and authenticates it using TWPubKey.
5. Fetches the BL30 Content certificate and authenticates it using the BL31 key certificate.
6. Fetches the BL32 image and verifies it against the hash of the content certificate.
7. Fetches the UEFI boot public key (UEFIPubKey – dbb) from the UEFI variable store.
8. Fetches the BL33 public key hash and authenticates it using the dbb.
9. Fetches the BL33 (UEFI) image and authenticates it using the BL33 public key.

After successfully authenticating BL31 and BL33, the primary socket logical core 0 runs ATF BL31. If any preceding step fails authentication, the ATF BL2 Firmware halts the boot process and communicates with the SMpro firmware. The BMC can query the SMpro for any boot failure condition and log an appropriate event.

After the ATF BL31 and BL32 Firmware finish initializing, the primary socket logical core 0 runs BL33 (UEFI), exiting the secure world using an ERET instruction to jump to EL2.

10.11 Unified Extensible Firmware Interface (UEFI) Secure Boot

The UEFI firmware continues execution according to the UEFI secure boot specification. Optional ROM or external UEFI modules, such as PCIe ROM or an OS boot loader shim, are authenticated. UEFI secure boot uses these key databases for authentication:

- Platform Key (PK).
- Key exchange key (KEK).
- Allow Database Key (DB).
- Disallow database key (DBX).

10.11.1 Platform Key (PK) Database

As described in the UEFI specification, a PK establishes a trust relationship between the platform owner and the platform firmware. The platform owner enrolls the public half of the PK (PKpub) into the platform firmware. The platform owner can later use the private half of the PK (PKpriv) either to change platform ownership, or to enroll a KEK.

10.11.2 Key Exchange Key (KEK) Database

A KEK establishes a trust relationship between the OS and the platform firmware. Each OS (and, potentially, each third-party application that must communicate with the platform firmware) enrolls a public KEK (KEKpub) into the platform firmware.

10.11.3 Allow Database Key (DB)

The DB database includes a key for authenticating optional and external modules.

10.11.4 Disallow (DBX) Key Database

The DBX database includes keys which must be invoked.

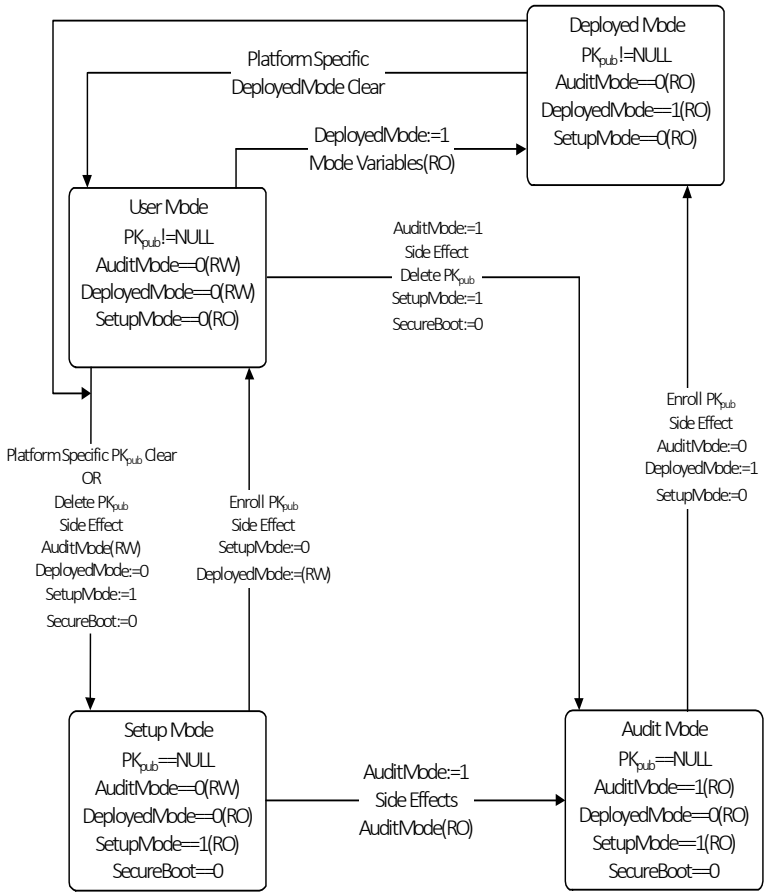
The relationship between the listed keys follows:

```
Platform Key (PK) --sign--> Key Exchange Key (KEK) --sign--> Allow DB (DB)
                                                   --sign--> Disallow DB (DBX)
```

These keys are *not* stored as UEFI persistent environment variables. On the Altra Max platform, these keys are accessed using the Altra Max custom ATF BL31 Secured Monitor Call (SMC). This SMC interface is described in this document.

Figure 10-3 shows the UEFI secure boot operating modes.

Figure 10-3: UEFI Secure Boot Operating Modes



10.12 Chain of Trust (CoT)

A Chain of Trust (CoT) starts with a set of implicitly trusted components. On the Ampere Computing development platforms, these components are:

- The hashed root public key stored in eFuses.
- The untampered ROM image.

The remaining components in the CoT are second-level certificates and firmware/bootloader images. The second-level certificates and firmware/bootloader images are signed.

The system uses Azure Key Vault (AKV) to generate all keys and certificates. AKV also stores the Signer Secret Key and performs signing processes. When a new key or certificate is generated, the request is logged and emailed to Ampere Computing stakeholders for auditing.

10.12.1 Certificate Hierarchy

There are separate signing/authentication domains, as shown in *Figure 10-4*, with a new root of trust

- Ampere signed images/certificates for SMpro and PMpro firmware, ATF, and associated certificates.
- Unsigned provisioned structures, such as UEFI variables and the NVRAM partition, which are measured.
- If enabled, customer/OEM signed images/certificates, such as UEFI.

Figure 10-4 illustrates the key and certificate signing and authentication domains.

Figure 10-4: Signing/Authentication Domains

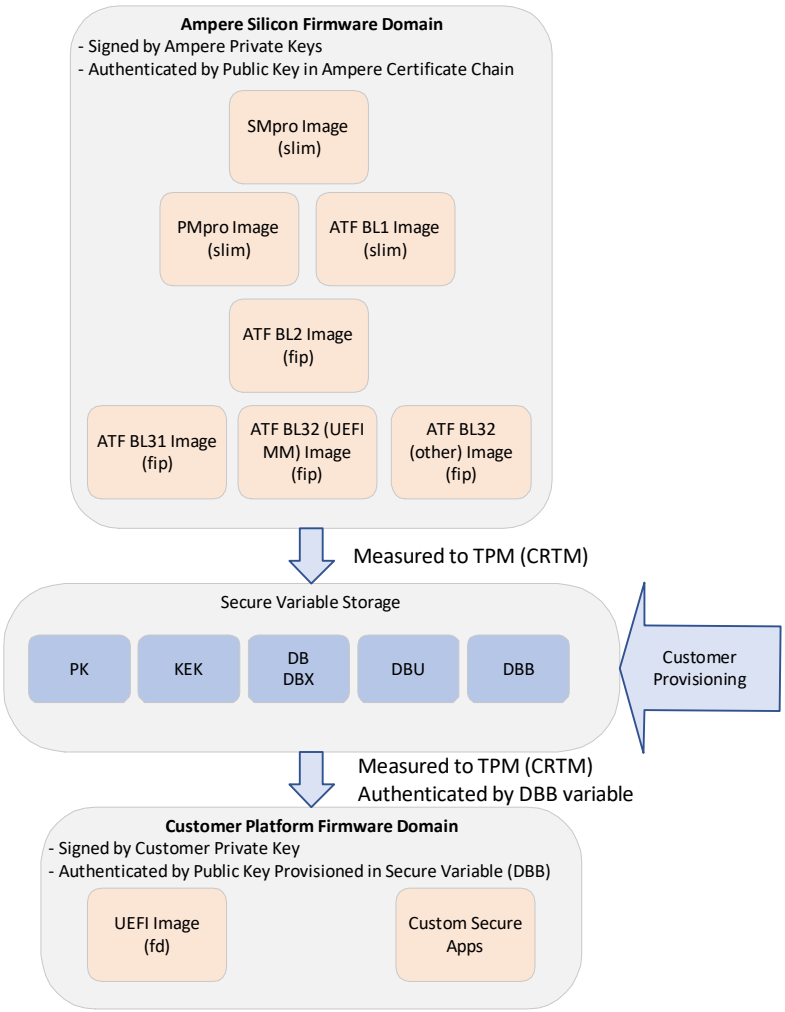


Figure 10-5 illustrates the key, certificate, and image hierarchy and certificate domains

Figure 10-5: Key, Certificate, and Image Hierarchy and Flows

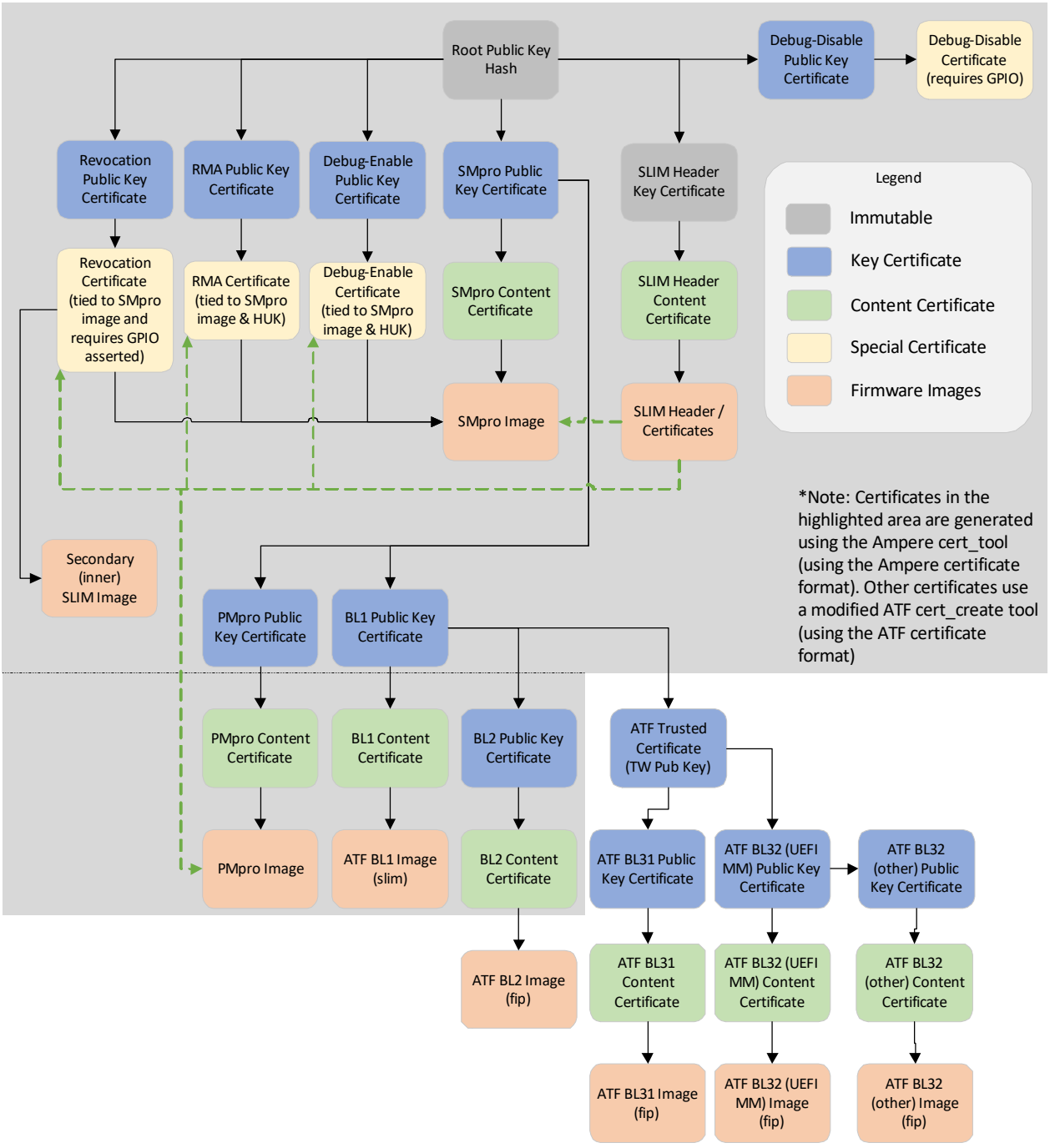
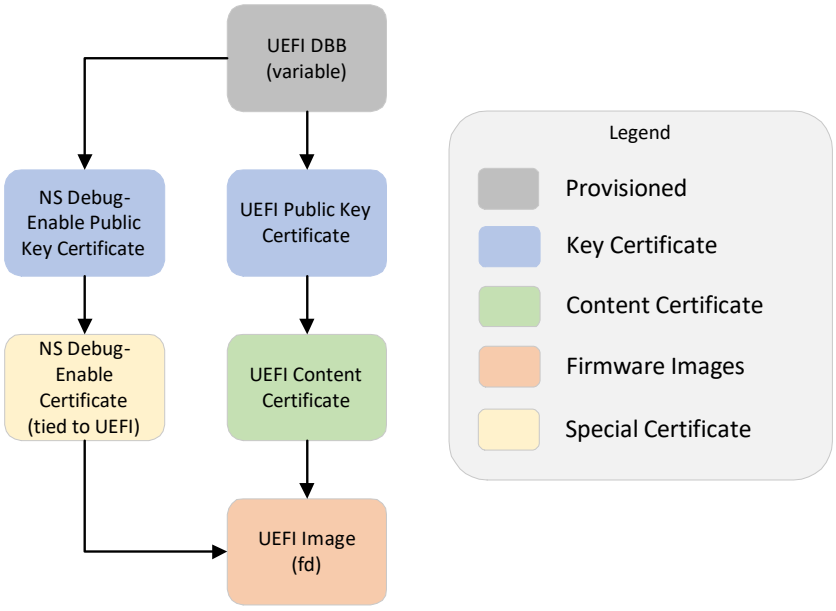


Figure 10-6 illustrates the UEFI key and certificate hierarchy.

Figure 10-6: UEFI Key and Certificate Hierarchy



10.12.2 Two-Level Signing and Authentication

The root key pair and second-level key pair, both generated using OpenSSL tools and implementing the Rivest–Shamir–Adleman (RSA) algorithm, generate the root key pair and second-level key pair. The private key in the root key pair signs the second-level public key and creates the key signature. In turn, the private key in the second-level key pair signs the image and creates the image signature. The second-level key signature and the image signature, along with the second-level root key, are all used to authenticate the image.

The Altra Max secure boot implementation uses two-level key authentication to maintain the CoT from ROM to the ATF BL31 runtime. Users can also register their own UEFI certificate to maintaining a secure CoT from UEFI to OS startup.

Two-level key authentication is implemented in these steps:

1. The second-level public key signature is verified using the root public key. The output of the decryption operation is a 256-bit hash value. This hash value is then compared with the calculated hash value of the second-level public key. The second-level key is authenticated if both hash values match.
2. The image signature is verified using the second-level public key that was authenticated in the preceding step. The generated hash value of the decryption operation must match the calculated hash value of the image to ensure that the image is not tampered.

If either step fails, the image is not authenticated.

10.13 Secure Boot Keys and Certificates

Table 10-1: Secure Boot Keys and Certificates (Sheet 1 of 2)

CERTIFICATE	PROVIDER	COMMENTS
ROM root public key Hash	Ampere Computing	This public key hash is either in the ROM as part of the SoC manufacturing process for production parts, or can be retrieved from the TMM eFuse (if ROM key hash is revoked).
SMpro public key certificate	Ampere Computing	This public key certificate is signed with the ROM Root private key.
PMpro public key certificate	Ampere Computing	This public key certificate is signed with the SMpro private key.
ATF BL1 public key certificate	Ampere Computing	This public key certificate is signed with the SMpro private key.
ATF "Trusted Certificate"	Ampere Computing	This certificate is signed with the ATF BL1 private key.
Trusted World public key (TWPubKey)	Ampere Computing	This public key is contained in the ATF "trusted certificate," which is signed by the ATF BL1 private key.
Normal World public key (NWPubKey)	N/A	This public key is contained in the ATF "trusted certificate", which is signed by the ATF BL1 public key. This key is not used because the trusted certificate is signed by Ampere, while the UEFI image must be signed by the platform vendor. To authenticate the UEFI image, the platform vendor must use the "UEFI boot policy key."
DBB (UEFI boot policy key)	Platform vendor	ATF uses this public key to authenticate the UEFI image during boot. To provision this key, the user sets an authenticated UEFI variable using this Globally Unique Identifier (GUID) and name: {0x4796d3b0, 0x1bbb, 0x4680, {0xb4, 0x71, 0xa4, 0x9b, 0x49, 0xb2, 0x39, 0x0e}} dbb. **Note**: Before provisioning this key, the platform vendor must enable UEFI secure boot as described in *Section 10.11, "Unified Extensible Firmware Interface (UEFI) Secure Boot."*
DBU (UEFI update key)	Platform vendor	ATF uses this public key (sometimes called the "production key") to authenticate UEFI update capsules during firmware update. To provision this key, the user sets an authenticated UEFI variable using this GUID and name: {0x4796d3b0, 0x1bbb, 0x4680, {0xb4, 0x71, 0xa4, 0x9b, 0x49, 0xb2, 0x39, 0x0e}} dbu. **Note**: Before provisioning this key, the platform vendor must enable UEFI secure boot as described in *Section 10.11, "Unified Extensible Firmware Interface (UEFI) Secure Boot."*
BMC update image key	Platform vendor	The same key should be used to sign the SMpro, ATF/UEFI, and BMC images. For some platforms, this may be the same key as the "Update Key." **Note**: BMC update key provision varies depending upon the BMC solution.

Table 10-1: Secure Boot Keys and Certificates (Sheet 2 of 2)

CERTIFICATE	PROVIDER	COMMENTS
PK (pk)	Platform vendor	
KEK (kek)	Microsoft	
DB (db)	Microsoft	
DBX (dbx)	Microsoft	

10.14 Certificate Formats

Note: Certificate formats are under development. The contents of tables in this section may change.

Table 10-2: Certificate Extension Structure (Sheet 1 of 2)

FIELD	BYTE LENGTH	BYTE OFFSET	DESCRIPTION
Major Revision	1	0	This value indicates the major revision of the structure; must be set to 1.
Minor Revision	1	1	This value indicates the minor revision of the structure; must be set to 0.
IANA Number	2	2	This value must be set to 0xCD3A, the Internet Assigned Numbers Authority (IANA) Number assigned to Ampere Computing.
Software Revision	4	4	This value is associated with the anti-rollback software version that is stored in the TMM eFuse. For SMpro/PMpro images; this is associated with the PRIMARY_VERSION in the TMM eFuse. **Note**: Only the SMpro key and content certificate may have a security version greater than what is in the PRIMARY_VERSION fuses. If this is true, the SMpro firmware updates the fuses. All other SMpro/PMpro images/certificates must have a security version matching the fuses (and therefore matching the SMpro certificate security version). For ATF images: This is associated with the SECONDARY_VERSION in the TMM eFuse. **Note**: Only ATF BL1 key and content certificate may have a security version greater than what is in the SECONDARY_VERSION fuses. If this is true, the SMpro firmware updates the fuses. All other ATF images/certificates must have a security version matching the fuses (and therefore matching the ATF BL0 Certificate security version).

Table 10-2: Certificate Extension Structure (Sheet 2 of 2)

FIELD	BYTE LENGTH	BYTE OFFSET	DESCRIPTION
HUK	32	8	The Hardware Unique Key (HUK) field provides a unique device ID. If zero: Indicates that this certificate is not associated with any unique device. If non-zero: Indicates that this certificate is to be associated with the unique device ID populated in this field, which maps to the manufacturing fuse values Bytes 16:0 – ECID Byte 17 – SPEED_GRADE_FREQ Byte 18 – VERSION Byte 19 – PART_NUMBER Byte 32:20 – Must be set to zero **Note:** For specific certificate types, a value of zero is considered invalid and results in authentication failure.
Image hash	32	40	This value is the 256-bit hash value for the image associated with this certificate. For certificates that are not associated with another image (for example, Debug-Disable Content Certificate), the value of this field must be set to zero.
Certificate subtype	1	72	Setting the value of this field to one of these values indicates the certificate type: 1 Normal-Boot (extension fields described in *Table 10-3*) 2 Debug-Enable (extension fields described in *Table 10-4*). 3 Debug-Disable (extension fields described in *Table 10-5*). Other values are reserved and are considered invalid for Altra Max.
Certificate subtype length	1	73	This value indicates the length of the certificate subtype-specific structure in bytes. For Altra Max, this must be set to 40.
Reserved	2	74	Must be set to zero.
Certificates subtype-specific structure	68	76	See *Table 10-3*, *Table 10-4*, and *Table 10-5* for details.

10.14.1 Normal-Boot Extension Fields

Table 10-3: Normal-Boot Extension Fields

FIELD	BYTE LENGTH	BYTE OFFSET	DESCRIPTION
Secondary Image Hash	32	0	This contains the 256-bit hash value for the secondary image (e.g. SMpro) associated with this certificate.
Reserved	292	32	Must be set to zero.

10.14.2 Debug-Enable Extension Fields

Table 10-4: Debug-Enable Extension Fields

FIELD	BYTE LENGTH	BYTE OFFSET	DESCRIPTION
Debug Enable Mask	1	0	This value indicates the secure debug re-enable mask. This value in this field is associated with the debug enable mask: Bit 0: AuthSpiden (enables secure privileged invasive CPU/CMI debug) Bit 1: AuthSpniden (enables secure privileged noninvasive CPU/CMI debug) Bit 2: AuthSmpro (enables SMpro invasive debug) Bit 3: AuthPmpro (enables PMpro invasive debug) Other bits are reserved and must be set to zero.
Reserved	67	1	Must be set to zero.

10.14.3 Debug-Disable Extension Fields

Table 10-5: Debug-Disable Extension Fields

FIELD	BYTE LENGTH	BYTE OFFSET	DESCRIPTION
Debug Disable Mask	1	0	This value is associated with the debug control mask in the TMM eFuse: Bit 0: DBGDIS; disables invasive debug Bit 1: SPIDEN; enables secure privileged invasive debug. Bit 2: NIDDIS; disables non-invasive debug. Bit 3: SPNIDEN; enables secure privileged non-invasive debug. Other bits are reserved and must be set to zero.
Reserved	67	1	Must be set to zero.

10.15 Dual-Socket Platform (2P) Authentication Flow

This is a simplified description of the 2P authentication flow, which is described in *Section 10.16, "Secure Boot Storage."*

1. Perform Elliptic-Curve Diffie–Hellman (ECDH) based key exchange; a unique symmetric key is generated for every boot.
   - Each socket generates a random number using a Pseudo-Random Number Generator (PRNG).
   - Each socket generates a "public value" using a "shared secret seed" in eFuses.
   - The primary and secondary sockets exchange the "public value"; this can be done over I^2C or untrusted CCIX.

     In ECDH, a "public value" is used with a "shared secret seed" to generate a unique "private key" that can then encrypt communications.
2. After each socket derives a secret per-boot symmetric key, the resulting secure communication channel can begin exchanging additional information for subsequent authentication.
   - The primary and secondary sockets securely exchange signed and encrypted hashes of the secure fuse configuration; this can be done over I^2C or untrusted CCIX.
     - The primary socket validates the secondary fuse hash.
     - The secondary socket validates the primary fuse hash.
     - The primary socket logs the secondary device hash to TPM.
   - The primary and secondary sockets securely exchange these signed and encrypted hashes; this can be done over I^2C or untrusted CCIX:
     - SMpro firmware.
     - PMpro firmware.
     - ATF BL1 Firmware.
   - Platform configuration data and the boot variable storage consumed by each socket.

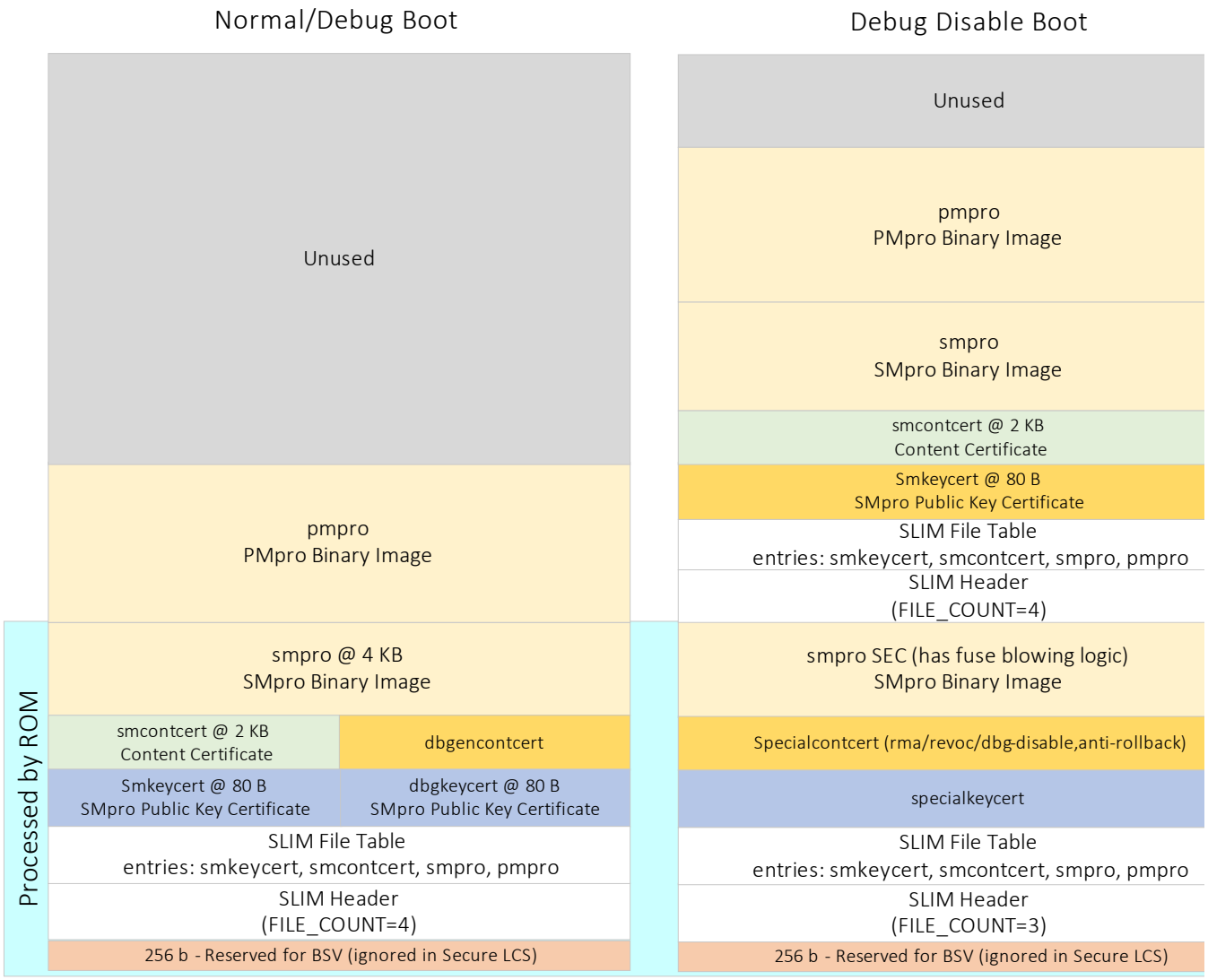
10.16 Secure Boot Storage

The Altra Max secure boot solution has a hard requirement for an I^2C EEPROM containing a minimum of 256 KB that must be attached to the SMpro I^2C bus. In addition, the boot solution requires a SPI-NOR flash to be connected on the SPI bus with a minimum size of 32 MB.

10.16.1 EEPROM Layouts

Figure 10-7 shows the EEPROM layouts for a) normal boot mode and b) normal debug disable boot mode.

Figure 10-7: EEPROM Layouts in Normal/Debug Boot and Normal Debug Disable Boot Modes



10.16.2 SPI-NOR Storage

The SPI-NOR storage contains ATF, delivered by Ampere Computing, along with the UEFI partitions, owned by the platform builder. The UEFI partitions include the UEFI image, UEFI secure variable storage, UEFI certificates (if UEFI authentication is enabled), and an NVRAM partition for storing platform configuration data to be consumed by SMpro firmware/ATF.

Power Management

11

This chapter describes power management features and their use in Altra Max processors. The chapter covers these topics:

11.1 Overview

Altra Max processor power management features improve its energy efficiency, which in turn enhances processor performance. Each processor block implements various power states. Some states are entered automatically; that is, they are not directly controlled by the user. Users can select other states through control registers.

The on-chip PMpro power management microcontroller exposes and supports ACPI features, such as LPI and CPPC.

Altra Max processor power management features include:

- ACPI firmware interface.
- Control of an external VRM to support DVS for CPMs.
- DFS of CPMs.
- Adaptive hardware control of voltage and frequency to maximize performance for a given power level.
- Dynamic power estimation.
- DDR per-rank self-refresh.
- Hardware-initiated power-saving states in the MCUs.
- Numerous temperature sensors and counters for power events.
- Voltage-drop mitigation; hardware detects rare, large, temporary voltage drops and takes actions to compensate.

An Altra Max processor has two primary power domains that are separate from each other and from a few additional secondary power domains. The primary power domains, in the order in which they are powered up, are the:

- SoC power domain (MCUs, GIC, SMpro and PMpro, and SoC peripheral interfaces).
- PCP power domain (CPMs and the CMI).

An Altra Max processor connects to an external power control device that drives the power supplies and informs the processor when the power supplies for both power domains are stable.

"Power management" is a collective term for the numerous devices, interfaces, and techniques used to optimize not only power consumption, but also processor performance. For Altra Max processors, these power management components and techniques handle power management tasks:

- Idle Management:
  - Core idle management.
  - System idle management.
- Performance management:
  - Process Voltage Scaling (PVS), sometimes called Adaptive Voltage Scaling (AVS).
  - DVFS.
- Limit management:
  - Thermal limits.
  - Power limits.
- Droop Mitigation to reduce the quantity and effect of VDD noise.
- Frequency slewing to reduce di/dt noise events.

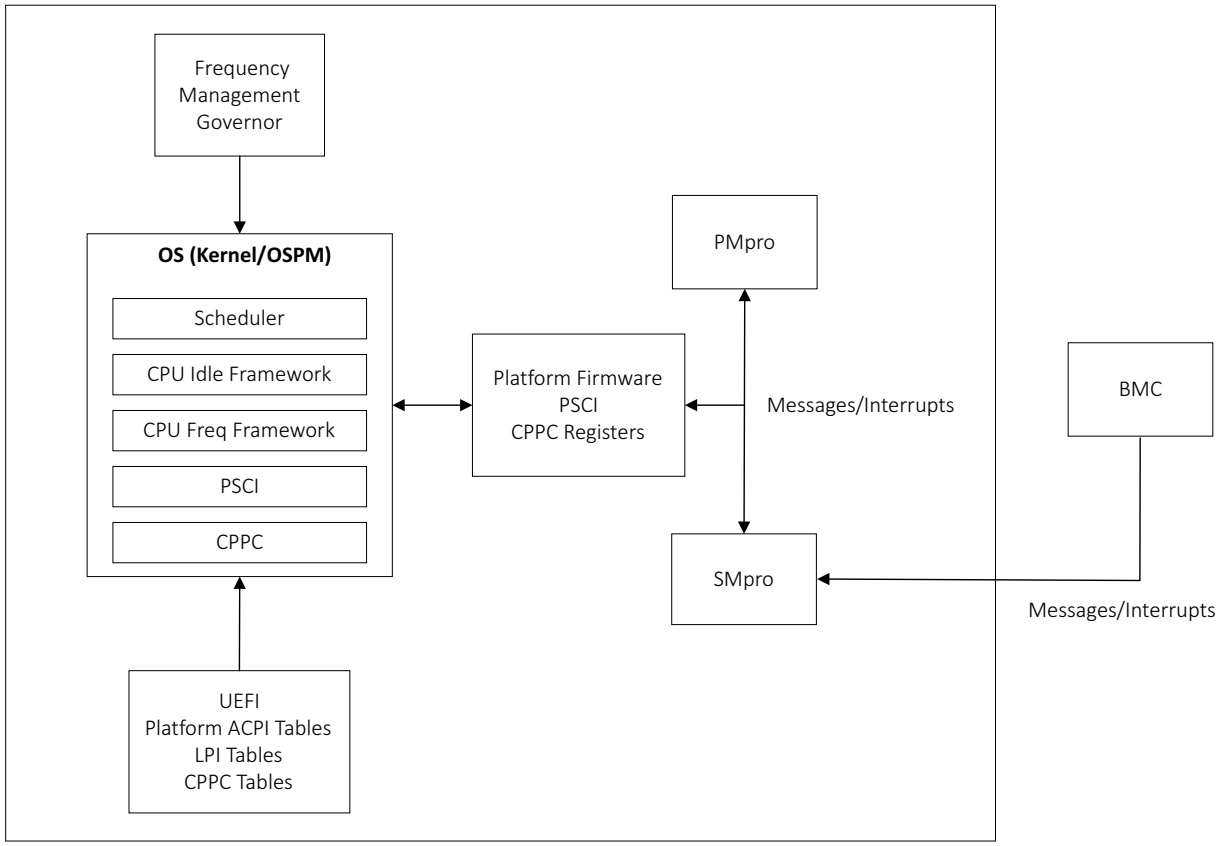
For detailed power measurements, including Thermal Design Power (TDP), refer to *Altra Max Datasheet*.

11.1.1 Power Management Entities

On a platform, various entities perform power management functions in collaboration, from the BMC controlling power sequencing and platform power management to the OS managing requests from the Operating System-Directed Configuration and Power Management (OSPM) ACPI component.

Figure 11-1 illustrates these entities and their relationships. Except for PMpro and SMpro, these entities are external to the Altra Max processor.

Figure 11-1: Power Management Entities



Each entity performs a particular function in managing power on a platform:

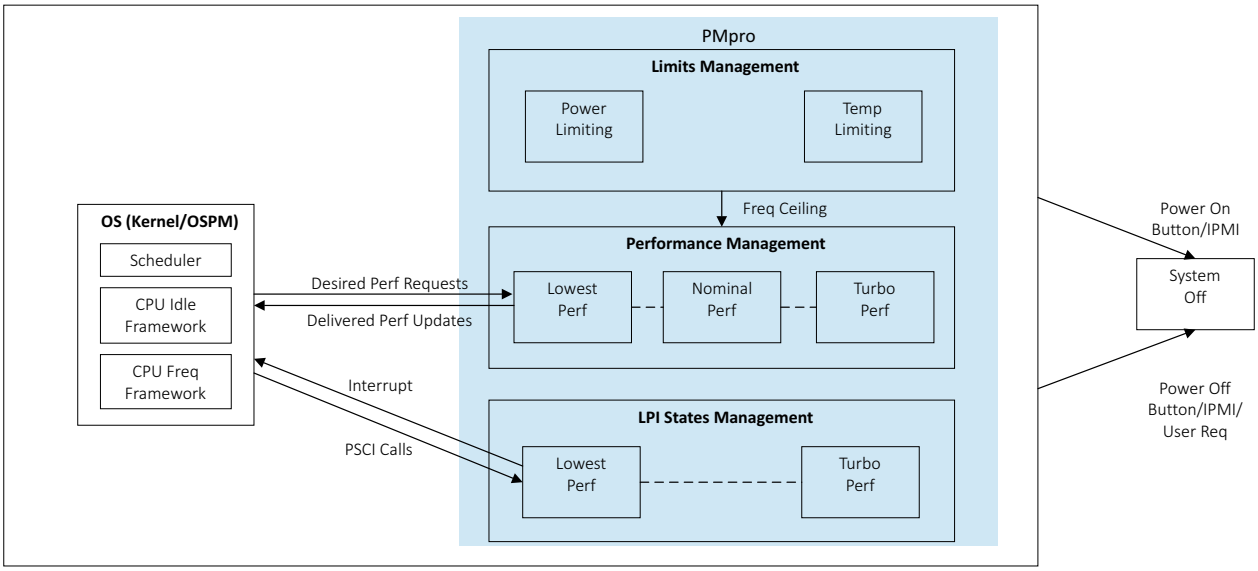
- OS (Kernel/OSPM):
  - Performs system-level power management.
  - Selects core and system idle states (LPI) and manages performance based on frequency management governor settings.
- UEFI ACPI LPI and CPPC tables:
  - Describe platform capabilities to the OS.

- PSCI and CPPC registers (platform firmware):
  - Trap calls from the OS to handle platform-specific implementations.
  - Communicate with power management functions (PMpro), for example, informing when a core goes idle, or requesting a core to be turned on or off (PSCI calls CPU_ON or CPU_OFF).
- PMpro:
  - Main platform power manager.
- SMpro:
  - Provides a path to the BMC for platform power management (power-off requests, power sequencing, interrupts, and so on).
- BMC:
  - Supervises platform power management.

11.1.2 Power Management Functions

Figure 11-2 illustrates the relationships among an Altra Max processor (or processors, in 2P systems), platform components, and the power management functions.

Figure 11-2: Power Management Functions



PMpro handles platform power management, which comprises three domains: limits management (power and thermal limits), performance management (managing core frequencies, for example) and power savings measures (LPI states management).

- LPI states management enables an OS to selectively turn cores on and off and manage core idle states based on processor workloads.

  Section 11.3, "Idle Management," describes this domain.
- Performance management determines the processor performance level, in particular core frequencies, based on processor capabilities, input from limits management, and OSPM requests.

  Section 11.4, "Performance Management" and *Section 11.5, "Performance Feedback Counters"* describe this domain.

- Limits management ensures that processors perform in safe and guaranteed power consumption ranges and thermal envelopes. This function manages processor power to keep it under TDP to avoid undue stress on the platform power delivery system, and monitors temperatures to prevent thermal damage to the processor.

  Section 11.6, "Limits Management," describes this domain.

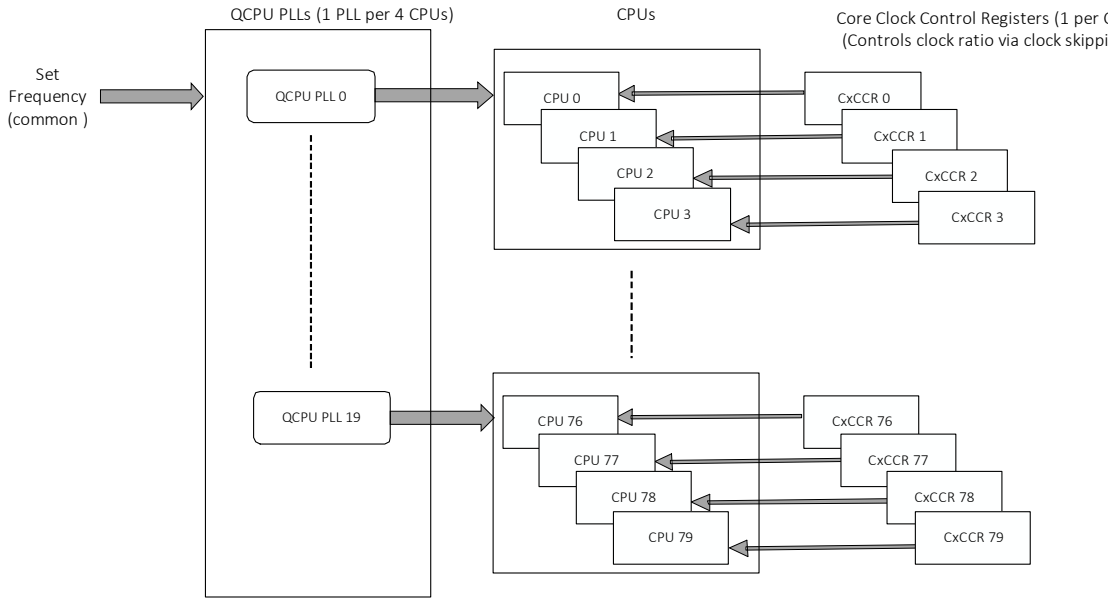
11.1.3 Altra Max CPU Clock Architecture

Power management functions must be able to modify core frequencies to manage power consumption, temperature, and overall performance. For more information about Altra Max processor clocks, see *Chapter 9, "Clocks and Reset."*

Core frequency is a function of clocks provided by QCPU PLLs and clock cycle skipping.

An Altra Max processor provides 32 QCPU PLLs. Each QCPU PLL delivers a clock to a group of four separate cores. Clock cycle skipping, configured in one or more CxCCR registers, can reduce core frequency from the QCPU PLL frequency. Each core has its own CxCCR register that enables the four cores sharing a QCPU PLL to have their clocks adjusted independently.

Figure 11-3: Altra Max CPU Clock Architecture



11.2 Power Management Infrastructure

The Altra Max power management infrastructure is primarily built on the PMpro power management microcontroller and ACPI, particularly on the Operating System-Directed Configuration and Power Management (OSPM) ACPI component. Using firmware running primarily on PMpro, ACPI and OSPM enable software control of important power parameters.

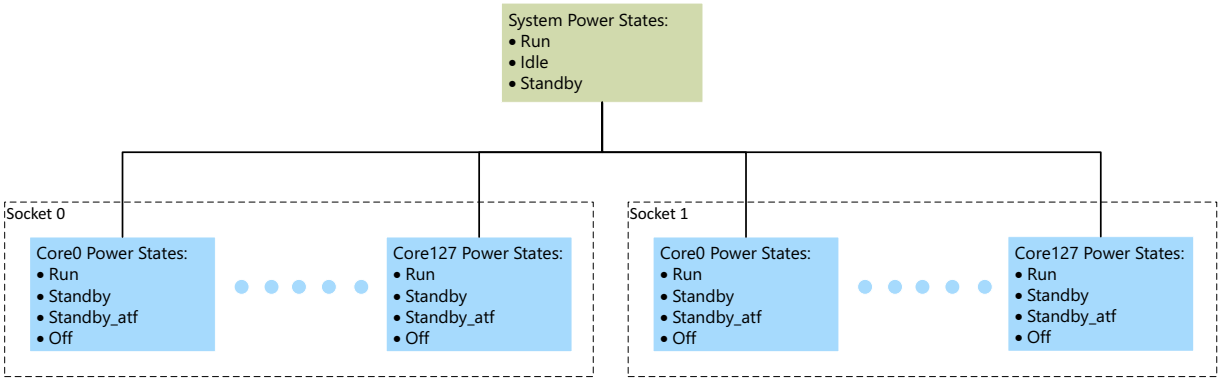
ACPI provides an abstract layer of platform-independent power management functions that enable voltage and frequency controls for major Altra Max processor and system components. In this context, the term "system" refers to a 1P or 2P server

platform. In *Figure 11-4*, the power states listed in the box labeled "System Power States" apply to this level. The power states listed in the boxes labeled "Core*n* Power States" apply to the core level.

Note: Both ACPI and OSPM support additional hierarchical levels and power states. *Figure 11-4* shows the hierarchy and power states supported in Altra Max systems.

After an OSPM-compatible OS activates ACPI, the OS takes exclusive control of all aspects of power management and device configuration. The OSPM implementation must expose an ACPI-compatible environment to device drivers, which exposes certain system, device and processor states.

Figure 11-4: Power State Hierarchy



Note: The Altra Max LPI states do not include "cluster" level power states. In a 2P system, an Altra Max processor is never put into a reduced PCP clock state while the other Altra Max processor has active cores and accesses DDR across the CCIX interface between the two sockets.

The power management infrastructure includes additional devices:

- Performance feedback counters for core performance monitoring.
- A network of thermal sensors to manage thermal limits.
- A network of Power Event Monitors (PEMs) to manage power limits.

11.3 Idle Management

Altra Max processors implement the ACPI 6.1 specification for idle management. This specification defines the core and systems idle power states shown in *Figure 11-4*.

11.3.1 System-Level Idle Management

If at least one core is running on the Altra Max processor in a 1P system, or on either Altra Max processor in a 2P system, the system is in the Run system power state. When the last core goes idle, it notifies PMpro, which places the system into one of the low-power system idle states. The selected power state depends upon the low-power state (Idle or Standby, as described in *Table 11-1*) that OSPM requests for the system.

Table 11-1 summarizes the supported system power states, which are described in more detail following the table. In the table, Quad CPU (QCPU) refers to a set of four CPU cores that share a clock and its corresponding PLL.

Table 11-1: System Power States

SYSTEM POWER STATE	LPI STATE	PCP VOLTAGE	SoC STATE	CORE CLOCK	CORE0 STATE	CORE*n* STATE	QCPU PLL	DRAM
Run	LPI-0	On	Active	On	At least one core is running.		On	Active
Idle	LPI-1/LPI-2	On	Active	Gated	Standby/Standby_atf/Off		On	Active
Standby	LPI-3	On	Active	Gated	Standby_atf/Off		On	Self-refresh

11.3.1.1 System Run State

In the system Run state, at least one core is in the Run state. Other cores can be in the Standby, Standby_atf, and Off core power states.

11.3.1.2 System Idle State

In the system idle state, all cores are in the Standby core power state or a lower power state (Standby_atf or Off states). This system Idle state is implicit when all cores enter the Standby state through WFI instructions in the OS.

All QCPU PLLs remain at full speed, and clock skipping using the CxCCR registers is unavailable. However, clocks are gated in each core. IOs and paths to DRAM remain available, all QCPU PLLs remain at full speed, and clock skipping using CxCCR registers is not supported. However, clocks in each core are gated. The PCP and SoC power domains remain fully powered.

The System Idle State is not an LPI state and is not exposed as such in ACPI tables.

11.3.1.3 System Standby State

All cores in the system standby state are in the Standby_atf core power state. This state is entered when all cores enter the Standby_atf state through OSPM calling the CPU_SUSPEND function. DRAM is put into self-refresh. The system is inactive but is immediately available for active operation when the DRAM exits self-refresh. The system still responds to interrupts as wake events to PMpro to exit DRAM from self-refresh.

All cores are in the Standby_atf core power state at their highest state; some cores may be in the Off state. The system Standby state is implicit when all cores enter the Standby_atf state through WFI instructions in the OS.

In system Standby state, QCPU PLLs run at reduced speeds. Clocks in the cores are gated.

The System Standby State corresponds to the LPI-3 state and can be exposed as such in the ACPI tables.

The system standby state typically has a fairly high minimum residency and worst-case wake-up latency, much higher than the core Standby and Standby_atf states (see *Section 11.3.2, "Core-Level Idle Management"*).

11.3.2 Core-Level Idle Management

When an OS kernel has no thread to schedule on a core, the OS asks OSPM to place that core into a low-power idle state. The OSPM places the processor core into one of the supported power states. The selected power states depends upon how quickly the core will be needed again. A core executes a WFI instruction to enter a low-power state.

Table 11-2 summarizes the processor core power states, which are described in more detail following the table.

Table 11-2: Core Power States

POWER STATE	VOLTAGE	STATE	DESCRIPTION
Run	On	Active	Active execution.
Standby	On	Wait	Not last core wait state, entered through executing WFI.
Standby_atf	On	Wait	Last core wait state, entered through executing WFI.
Off	On	Off	Off state; architected core state is lost.

11.3.2.1 Core Run State

In the Run state, a core is powered up and running code.

11.3.2.2 Core Standby State

This is the default core low-power state. A core executes a WFI or WFE instruction in user mode code (EL1) to enter the Core Standby state.

In the core Standby state, a core is powered but does not run code. The core clock is gated at the root of the clock tree. The core Standby state maintains core context. An external debugger can access debug registers in the core power domain.

The core Standby state corresponds to the LPI state LPI-0 and typically has the lowest minimum residency and low worst-case wake-up latency. This enables cores to quickly return to the Run state with minimal impact on performance.

11.3.2.3 Core Standby_atf State

OSPM can select the Standby_atf low power state for any core in the system that goes idle. When the last core goes idle, OSPM also select an idle state for the system. OSPM calls the CPU_SUSPEND function to enter the core Standby_atf state. This function sends a message to all relevant levels of execution. The exception level with the highest privilege (EL3) executes a WFI instruction after running this sequence:

1. Configure the system timer for wake-up event.
2. Notify PMpro.
   a) Write LPIReqn registers.

In the core Standby_atf state, the core is powered up but does not run code. The core clock is gated at the root of the clock tree. The core Standby_atf state maintains core context. An external debugger can access debug registers in the core power domain.

The core Standby_atf state corresponds to the LPI state LPI-1 and has low minimum residency and low worst-case wake-up latency. Both, however, are higher than for the core Standby state. This enables a core to still return quickly to the Run state, but the performance impacts of entering and exiting the core Standby_atf state are higher than for the core Standby state.

11.3.2.4 Core Off State

OSPM calls the CPU_OFF Function to dynamically remove a core from the system (hot plug). The OS migrates all interrupts and threads away from the core that is taken offline. The function sends a message to all relevant levels of execution. The exception level with the highest privilege executes a WFI instruction after running this sequence:

1. Clean cache.
2. Perform coherency management.

In the core Off state, architected core state is lost. An external debugger cannot access debug registers in the core power domain.

11.3.3 System Reset and System Shutdown

The Altra Max processor implementation of PSCI provides an interface that enables an OS to request system shutdown and system reset using these functions:

- SYSTEM_RESET.
- SYSTEM_OFF.

Table 11-3: PSCI Functions Supported in Altra Max Processors

PSCI FUNCTION	FUNCTION ID		DESCRIPTION
	SMC64	SMC32	
PSCI_VERSION	0x84000000		Returns the implemented PSCI version.
CPU_SUSPEND	0xC4000001	0x84000001	Suspends execution on a core or a higher layer topology node (such as a CPM or QCPU); used to enter low-power state.
CPU_OFF	0x84000002		Powers down the calling core.
CPU_ON	0xC4000003	0x84000003	Powers up a core.
AFFINITY_INFO	0xC4000004	0x84000004	Requests the status of an affinity instance.
SYSTEM_OFF	0x84000008		Shuts down the system.
SYSTEM_RESET	0x84000009		Resets the system.
PSCI_FEATURES	0x8400000A		Queries ACPI to discover whether a specific PSCI function is implemented, along with its features.

Note: PSCI specifies additional functions that are described in *ARM Power State Coordination Interface Platform Design Document*.

The Lower Power Idle States (LPI) extend to support the expression of idle states that are selected by OSPM when a processor goes idle, but which may affect more than one processor, and may affect other system components. LPI extensions in the

specification leverage the processor container device, and in this way can express which parts of the system are affected by a given LPI state.

11.3.4 Low Power Idle (_LPI) Table

_LPI is an object that provides a method to describe LPI states that defines the local power states for each node in a hierarchical processor topology. OSPM uses the _LPI object to select a local power state for each level of processor hierarchy.

This object may be used inside a Processor Container or a processor declaration.

Table 11-4: _LPI[n] Object Elements

ELEMENT	OBJECT TYPE	DESCRIPTION
Revision	Integer (WORD)	The _LPI.revision number.
LevelID	Integer (QWORD)	A platform-defined number that identifies the hierarchy level of the processor node to which the LPI states apply.
Count	Integer (WORD)	The count of subsequent LPI packages.
LPI[1]	Package	A package containing the definition of LPI state 1.
LPI[n]	Package	A package containing the definition of LPI state n.

Each LPI[n] subpackage contains the elements summarized in *Table 11-5*.

Table 11-5: _LPI[n] Subpackage Elements

ELEMENT	OBJECT TYPE	DESCRIPTION
Min Residency	Integer (DWORD)	Minimum residency time in µs.
Worst case wakeup latency	Integer (DWORD)	Worst case time in µs from a wake interrupt being asserted until returning to the running state of the owning hierarchy node.
Flags	Integer (DWORD)	Valid flags (1 if the power state is enabled for use).
Arch. Context Lost Flags	Integer (DWORD)	Architecture specific context loss flags.
Residency Counter Frequency	Integer (DWORD)	Residency counter frequency in cycles-per-second (Hz). Value 0 indicates that counter runs at an architectural-specific frequency.
Enabled Parent State	Integer (DWORD)	Index to the deepest local power state of the parent processor container _LPI enabled by this state.
Entry Method	Integer (DWORD)	An integer may be provided; if so, the integer is used to compose the final register value that must be used to enter this state.
State Name	String	String containing a human-readable identifier of the LPI state.

OSPM software can transition a core into any LPI state. Note that in the:

- Standby state: The core is put into WFI by the OS at EL1.
- Standby_atf state: The core is put into WFI at EL3 by ATF. Core and cache contexts are preserved.

- Off state: The core is put into the architectural "Power Off" state and core and cache states are lost. The exception level with the highest privilege cleans caches and performs coherency management before executing the WFI instruction. A core enters the Off state when it is taken offline. This results in the PSCI CPU_OFF command being sent to ATF for that core.

Table 11-6: System Power States at the LPI System Level

CORE POWER STATE	CORE STATES	QCPU PLL	PCP PLL	DRAM STATE	PCP DOMAIN	DESCRIPTION
Running	Run (at least one core)	Full speed	Full speed	Active	On	At least one core in the system (1P or 2P) is running
Standby	On	Reduced speed	Reduced speed	Self-refresh	On	The last core to enter Standby_atf (WFI-EL3) informs PMpro.

Note that:

- OSPM indicates whether a core going down in Standby_atf (WFI-EL3) is the last core down in the 1P or 2P system. This core signals its PMpro using LPIReqn registers; power management is coordinated between the sockets in a 2P system. In a 2P system, the PMpro for each socket takes its system-level resources into low-power states (clocks, voltages, memory, and so on).
- This is scalable and works in both 1P and 2P systems.

11.3.5 Lower Power Idle (LPI) State Change Implementation

The PSCI CPU_SUSPEND, CPU_OFF, and CPU_ON commands implement the LPI state changes.

11.3.6 Lower Power Idle (LPI) Core State Changes

All LPI state transitions are from Running to any other state or from any state to Running. A core can temporarily be in other states while transitioning to the final LPI state. For Altra Max processors, LPI state transitions occur in ATF or PMpro firmware.

11.3.7 Standby or Clock Gate State

The ACPI specification defines the Clock Gate state as a sleeping state that is implemented to reduce power consumption. The core executes a WFI instruction to enter this state and exits on a wakeup event.

11.3.8 Power Down State

What happens in this state depends upon the platform architecture and implementation. At the minimum, architectural context is lost because all cores share the same voltage rail and no dedicated controls are available to turn off power to a single core. The caches are cleaned and the architected state of the core is lost.

11.4 Performance Management

An Altra Max processor implements the ACPI v6.2 specification for power and performance management. PMpro software works with OSPM to support this functionality. A CPU in the running state can be put in any P-State to deliver the performance requested by OSPM.

11.4.1 Collaborative Processor Performance Control (CPPC)

CPPC controls CPU performance using an abstract continuous scale rather than a discrete P-State scale. If CPPC is present and enabled by OSPM, CPPC is used rather than Platform Communication Channel (PCC).

CPPC defines an abstracted and flexible mechanism for OSPM to collaborate with PMpro to manage the performance of a logical processor. In this scheme, PMpro creates and maintains a performance definition based on a continuous, abstract, and unitless performance scale. At runtime, OSPM requests a performance level on the abstract scale and PMpro translates the OSPM performance requests into actual hardware performance states. PMpro can also support the autonomous selection of performance levels appropriate for the current workload. In this case, OSPM conveys information to PMpro that guides performance level selection.

The control mechanisms are abstracted by the Continuous Performance Control (_CPC) object method, which describes how to control and monitor CPU performance in a generic manner. The register methods may be implemented in the PCC interface.

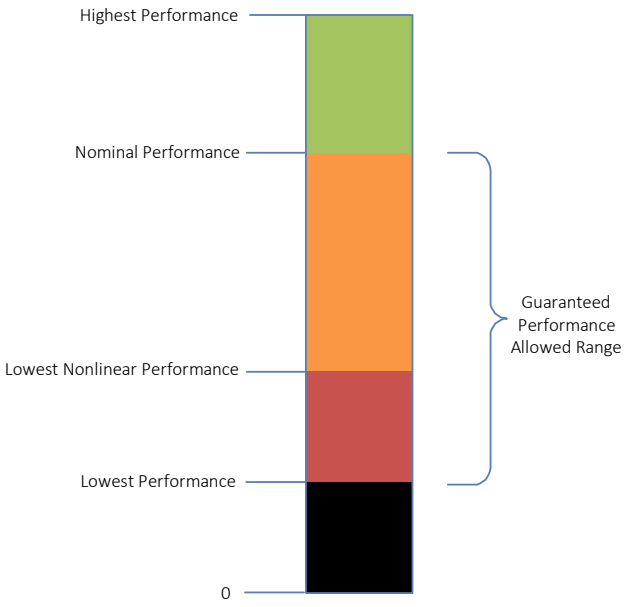
The _CPC object declares an interface that enables OSPM to transition the CPU into a P-State based on a continuous range of supported values shown in *Figure 11-5*; *Table 11-7* defines the associated CPPC performance thresholds.

Table 11-7: CPPC Performance Thresholds

THRESHOLD	DEFINITION
Highest Performance	The maximum performance an individual core can reach, assuming ideal conditions. This performance may not be sustainable on all cores simultaneously. For the Altra Max implementation, nominal performance and highest performance are identical.
Nominal Performance	The maximum sustained performance level of the processor, assuming ideal operating conditions. In absence of an external constraint (power, thermal, and so on), a processor is expected to continuously maintain this performance level. All cores are expected to sustain the nominal performance threshold simultaneously. For the Altra Max implementation, nominal performance and highest performance are identical.
Lowest Nonlinear performance	The lowest performance threshold at which nonlinear power savings are achieved, for example, from the combined effects of voltage and frequency scaling. Above this threshold, lower performance levels are generally more energy efficient than higher performance levels. In traditional terms, this represents the P-state range of performance levels.
Lowest Performance	The lowest performance level of the platform. Selecting a performance level lower than the lowest nonlinear performance threshold can actually cause efficiency penalties, but should reduce the instantaneous power consumption of the processor. On Altra Max, this is the minimum frequency at which a core can run; this typically occurs because of power or thermal constraints. In traditional terms, this represents the T-state range of performance levels.

The Guaranteed Performance Allowed Range shown in *Figure 11-5* is the supported range of guaranteed, sustained processor performance. This range considers known external constraints (power budgeting, thermal constraints, and so on). In this range, all cores are expected to sustain guaranteed performance levels simultaneously. This range includes frequencies in the CPPC Lowest Performance to Nominal Performance thresholds, inclusive.

Figure 11-5: CPPC Performance Thresholds



OSPM writes the desired performance value to the Desired Performance Register, and the platform maps the desired performance to an internal P-State. Performance control contains the elements (registers), described in *Table 11-8*, that control mechanisms defining how to control and monitor processor performance. Performance control exists under all CPUs. For the Altra Max implementation, nominal and maximum frequencies are identical.

Table 11-8: Altra Max CPPC Registers (Sheet 1 of 2)

ELEMENT	OFFSET	VALUE	DESCRIPTION
Num Entries	—	21	The number of entries int the _CPC package.
Revision	—	2	The Revision number of the _CPC package format.
Highest Performance	0x000	Maximum/Nominal Frequency	Indicates the highest level of performance the CPU can theoretically achieve (MHz). For the Altra Max implementation, nominal and maximum frequencies are identical.
Nominal Performance	0x004	Maximum/Nominal Frequency	Indicates the highest sustained performance level of the CPU (MHz). For the Altra Max implementation, nominal and maximum frequencies are identical.
Lowest Nonlinear Performance	0x008	Lowest Nonlinear Frequency	Indicates the lowest performance level of the CPU with nonlinear power saving (MHz).
Lowest Performance	0x00C	Lowest Frequency	Indicates the lowest performance level of the CPU (MHz).

Table 11-8: Altra Max CPPC Registers (Sheet 2 of 2)

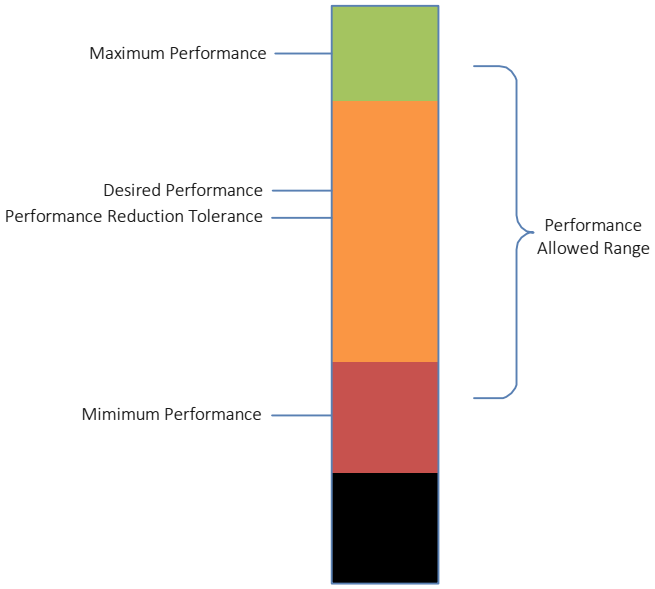
ELEMENT	OFFSET	VALUE	DESCRIPTION
Guaranteed Performance	0x010	Maximum/Nominal Frequency	For the Altra Max implementation, nominal and maximum frequencies are identical.
Desired Performance	0x014	Runtime assignment	OSPM writes the desired performance level (MHz).
Minimum Performance	—	N/A	—
Maximum Performance	—	N/A	—
Performance Reduction Tolerance	—	N/A	—
Time Window	—	N/A	—
Counter Wraparound Time	0x028	0	This value causes the counter to never wrap.
Reference Performance Counter	0x02C	Runtime accumulation	Counter accumulates at a rate proportional to the reference performance of the CPU.
Delivered Performance Counter	0x034	Runtime accumulation	Counter accumulates at a rate proportional to the delivered performance of the CPU.
Performance Limited	0x038	Runtime assignment	A non-zero value indicates performance is limited.
CPPC Enable	0x03C	True	—
Autonomous Selection Enable	—	N/A	—
Autonomous Activity Window	—	N/A	—
Energy Performance Preference	—	N/A	—
Reference Performance	0x04C	50 MHz.	Indicates the performance level at which the Reference Performance Counter accumulates.

Under CPPC, OSPM supports performance settings to control or influence platform performance, as shown in *Figure 11-6*.

The Highest Performance threshold and Guaranteed Performance Allowed Range inform the OS of available processor performance, the OS (and user) can use OSPM to set Minimum Performance and Maximum Performance levels to further constrain processor performance. The settings can be applied to provide the most appropriate performance for an application. This supports a user-defined Performance Allowed Range.

The Maximum Performance level can never exceed the Highest Performance threshold defined in *Table 11-7*, while the Minimum Performance level can never be lower than the Lowest Performance level defined in the same table.

Figure 11-6: OSPM Performance Settings



OSPM can select any performance value in the continuous range of values supported by the platform. Internally, the platform can implement a small number of discrete P-States and may not be able to operate at the exact performance level requested by OSPM.

To determine the actual performance level delivered over time, OSPM calculates the delivered performance over a time period using this equation:

$$Delivered\ \text{Performance} = \text{Reference performance} \times \frac{\Delta\text{Delivered performance counter}}{\Delta\text{Reference performance counter}}$$

11.4.2 Collaborative Processor Performance Control (CPPC) Implementation

11.4.2.1 Continuous Performance Control (_CPC) Table

The SoC _CPC table is implemented through the PCC interface, in PCC subspace 2, for a CPU. An example _CPC follows. Similar tables for the other CPUs have the same format, except for different register offsets.

```
Name(_CPC, Package()
{
    21, // NumEntries
    2, // Revision
    ResourceTemplate(){Register(PCC, 32, 0, 0x000, 2)}, //Highest Performance
    ResourceTemplate(){Register(PCC, 32, 0, 0x004, 2)}, //Nominal Performance
    ResourceTemplate(){Register(PCC, 32, 0, 0x008, 2)}, //Lowest Nonlinear Performance
    ResourceTemplate(){Register(PCC, 32, 0, 0x00c, 2)}, //Lowest Performance
    ResourceTemplate(){Register(PCC, 32, 0, 0x010, 2)}, //Guaranteed Performance
    ResourceTemplate(){Register(PCC, 32, 0, 0x014, 2)}, //Desired Performance
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, //Min Performance
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, //Max Performance
```

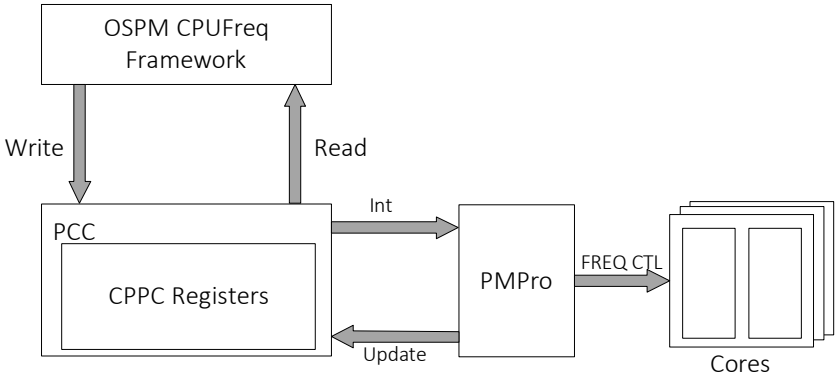
```
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, //Performance Reduction
Tolerance
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, //Time Window
    ResourceTemplate(){Register(SystemMemory, 64, 0, 0x024, 0)}, //Counter Wraparound
Time
    ResourceTemplate(){Register(PCC, 64, 0, 0x02C, 2)}, //Reference Counter
    ResourceTemplate(){Register(PCC, 64, 0, 0x034, 2)}, //Delivered Counter
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, // Performance Limited
Register
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, //Enable Register
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, //Autonomous Selection Enable
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, / Autonomous Activity Window
    ResourceTemplate(){Register(SystemMemory, 0, 0, 0, 0)}, //Energy Performance
Preference
    ResourceTemplate(){Register(PCC, 64, 0, 0x04c, 0)} // Preference Register
}) //_CPC
```

11.4.3 Altra Max Performance Management

In an Altra Max processor, PMpro sets the frequency of a QCPU PLL and then programs clock skip ratios to control delivered performance. Each QCPU PLL controls four CPU cores. An Altra Max processor contains up to 20 QCPU PLLs, which share a set of control registers. A Core Clock Control Register (CxCCR) for each core provides clock skipping logic that dynamically modulates the clock frequency for the core. The clock skipping rate is programmable in ratios of 1/32, with a range from 32/32 (no skipping) to 9/32. The combination of QCPU PLL frequency and clock skip ratio effectively scales the core clock frequency.

Communication between OSPM and PMpro occurs through PCC, a generic mailbox-like mechanism. The CPPC registers are implemented in PCC shared memory.

Figure 11-7: CPPC State Change Process



11.4.4 Using Platform Communication Channel (PCC) Registers

If the PCC register space is used, the CPPC registers must be in the same subspace. To write registers, OSPM fills in the register value and issues a PCC write command. To read static registers, counters, and the performance limited register, OSPM issues a read command. *Table 11-9* summarizes the CPPC commands.

Table 11-9: CPPC PCC Commands

COMMAND	DESCRIPTION
0x0	Read registers; executed to request PMpro to update registers for enabled CPUs with their current value.
0x1	Write registers; executed to notify PMpro that one or more RW registers or an enabled CPU was updated.
0x02 – 0xFF	Other values are reserved.

Note: In Altra Max processors, PCC is implemented using doorbell registers.

11.4.5 Collaborative Processor Performance Control (CPPC) Initialization Sequence

1. During boot, PMpro initializes the CPPC firmware driver.
2. UEFI creates the PCCT table inside the ACPI table that advertises PCC mailbox channel information.
3. UEFI initializes PCC mailbox shared memory to store the CPPC registers. This shared memory should store _CPC tables for all cores.
4. UEFI sends a CPPC read command to PMpro that enables PMpro to initialize the _CPC table for each core.
5. UEFI starts Linux.
6. Linux verifies and gets PCC mailbox channel information from the PCCT table.
7. If PCC is supported, Linux verifies and gets CPPC register information from the _CPC table.
8. If CPPC is supported, the Linux CPUFreq governor can use information in the CPPC registers to request a desired frequency during runtime based on a range of supported values.

Note: At boot time, all CPUs run at the Nominal Performance (Offset 0x004 of the _CPC table) CPPC State Change Implementation.

For Altra Max processors, the performance unit is core frequency and the frequency change affects one core.

This sequence describes what happens during a CPPC state change:

1. Linux makes a CPU frequency request, writing the desired value into the Desired Performance register (offset 0x014 of the _CPC table) for the core.
2. Linux sends a CPPC write command to PMpro.
3. PMpro receives the request and can change the QCPU PLL frequency setting and clock skip ratios.
4. PMpro periodically accumulates the delivered performance counter and reference performance counter.
5. PMpro returns the result to Linux using performance counters, which are used to determine the delivered CPU frequency.

Table 11-10: CPU Clock Skipping Ratio Range (Sheet 1 of 2)

CLOCK SKIPPING RATIO (CxCCR)	CORE FREQUENCY	DESCRIPTION
9/32	QCPU PLL frequency × (9/32)	Maximum clock skipping (minimum frequency)
10/32	QCPU PLL frequency × (10/32)	—

Table 11-10: CPU Clock Skipping Ratio Range (Sheet 2 of 2)

CLOCK SKIPPING RATIO (CxCCR)	CORE FREQUENCY	DESCRIPTION
....	....	—
32/32	QCPU PLL frequency × (32/32)	No clock skipping (maximum/nominal frequency)

11.5 Performance Feedback Counters

Each core provides activity monitoring, which has features used for performance monitoring. The activity monitors provide useful information for system power management and persistent monitoring. Each core implements five activity monitor counters. The activity monitors can be accessed by:

- Core system registers.
- Memory-mapped access using the debug APB interface.

Table 11-11: Activity Monitor Counter Register

NAME	WIDTH	DESCRIPTION
AMEVCNTR0_EL0	64	Delivered Performance Counter. This counter is incremented at the core clock frequency.
AMEVCNTR1_EL0	64	Reference Performance Counter. This counter is incremented at a constant frequency (system counter frequency).
AMEVCNTR2_EL0	64	Instructions architecturally executed. This counter increments for every instruction that is executed architecturally, including instructions that fail their condition code check.
AMEVCNTR3_EL0	64	The first miss event tracks whether any external load miss is outstanding and starts counting only from a first-miss until data returns for that miss. The counter does not count for any remaining part of overlapping accesses, but only counts again when the first-miss condition is detected again.
AMEVCNTR4_EL0	64	Instructions executing in the design that provide hints for potential high power activity.

To determine the actual performance level delivered over time, OSPM may read a set of performance counters from the Reference Performance Counter Register (AMEVCNTR1_EL0) and the Delivered Performance Counter Register (AMEVCNTR0_EL0).

OSPM calculates the delivered performance over a given time period, taking a beginning and ending snapshot of the reference and delivered performance counters, and calculating delivered performance using the formula found in *Section 11.4.1, "Collaborative Processor Performance Control (CPPC)."*

OSPM may use delivered performance counters as a feedback mechanism to refine desired performance state selection.

11.6 Limits Management

Limits management helps maintain operating constraints required by an Altra Max processor. The constraints limit the average maximum power consumed by the Altra Max processor and the instantaneous maximum processor or die temperature.

PMpro periodically monitors average Altra Max processor power consumption using:

- Voltage regulator telemetry over I^2C.
- PEMs distributed throughout the processor (see *Section 11.6.2.1.3, "Power Event Monitors (PEMs),"* for more information).

PMpro also periodically monitors maximum processor and die temperature using TSMs distributed throughout the processor, as described in *Section 11.6.1.2, "Thermal Sensors."*

Limits management is intended to:

- Keep maximum processor temperature under T_{cj} (typically 95 °C).
- Keep maximum average processor power under TDP or under an arbitrary limit imposed by the BMC or another external entity.

PMpro can scale CPU frequencies up to the maximum/nominal frequency based on OSPM requests. If PMpro detects that either the average processor power or maximum processor temperature exceed their respective limits, PMpro throttles core and CMI frequencies and scales down voltages as needed. This reduces both processor power consumption and die temperatures. After power and temperatures fall below their respective limits, PMpro enables the frequencies and voltages to gradually rise again.

11.6.1 Thermal Limits

Thermal sensors and thermal limit algorithms prevent Altra Max processor operation above the maximum junction temperature (T_{cj}) and shut down the processor when the temperature rises to a potentially damaging level. Thermal monitors are implemented using hardware (thermal sensors and GPIO ports) and software, implemented in firmware on PMpro. A software interface is provided to system software running on Altra Max cores. A set of hardware signals is provided to interface to other system components.

Altra Max processors provides two types of thermal limit algorithms:

- A local thermal limit management algorithm, which monitors only one core and is duplicated for every core (see *Section 11.6.1.3.2, "Local Thermal Limit Management Algorithm"*).
- A global thermal limit management algorithm, which monitors the entire processor (see *Section 11.6.1.4, "Global Thermal Limits Management"*).

11.6.1.1 Definitions

- T_{cj}: The maximum junction temperature for thermal design and sustained operation.
- $T_{HIGHTEMP}$: The temperature (100°C) at which Thermal Monitor 1 (TM1) is activated.
- $T_{OVERTEMP}$: The temperature (120°C) at which Thermal Monitor 2 (TM2) is activated.
- $T_{HIGHSENSOR}$: The value of the thermal sensor having the highest temperature reading.

- $T_{CORELIMIT}$: The temperature (by default, 95°C) at which the local thermal limit management algorithm is activated.

11.6.1.2 Thermal Sensors

The Altra Max processor provides a network of numerous thermal sensors for monitoring the die temperature. The power management firmware uses the sensor network, along with the power management and power monitoring features, to ensure that the operating temperature remains within the allowed range.

Thermal sensors are distributed across the Altra Max processor to cover the relevant components. Many components, including CPU cores, MCUs, PCIe RCs, and so on, have dedicated thermal sensors.

11.6.1.3 Local Thermal Limits Management

The $T_{CORELIMIT}$ temperature threshold is set for each core in the Altra Max processor. If a core temperature exceeds $T_{CORELIMIT}$, PMpro changes the CxCCR ratio to reduce the frequency for that core (see *Table 11-10*). Local performance reduction is the primary response to excessive temperature and is repeated as needed to reduce the core temperature.

The local thermal limit management algorithm does not directly affect core frequency; only DVFS or global thermal limits management algorithms directly affect frequencies in the processor. The local thermal limit management algorithm monitors core temperatures and provides additional data as input to the DVFS, which reduces core frequencies as needed.

11.6.1.3.1 DVFS Core Clock Management

To affect individual core frequencies, DVFS most often adjusts the Clock Ratio in the CxCCR registers (one per core) that uses clock skipping. DVFS can also adjust QCPU PLL frequencies, for example, when OSPM requires lower-than-maximum frequencies for multiple cores sharing a QCPU PLL. Because QCPU PLL changes affect at least four cores, independent, per-core frequency changes require adjusting the clock ratio in the corresponding CxCCR registers.

To determine which clock ratio to apply to a core, DVFS considers these parameters:

- CppcClkRatio.

  This clock ratio is required for the core to run at the frequency closest to an OSPM request. This is the main parameter and controls the clock ratio for typical runtime core activity, assuming no power or thermal limit is applied and the core is not in an LPI state.

- LtlmDeltaClockRatio.

  The local thermal limit management algorithm provides this parameter, which gives DVFS an adjustment to the final clock ratio used in CxCCR. When temperatures are excessive, LtlmDeltaClockRatio is higher than zero so that DVFS adds the number of skipped clock cycles equal to LtlmDeltaClockRatio to reduce core frequency.

  After the temperature returns to normal, LtlmDeltaClockRatio returns to zero because additional clock skipping is not required.

Note: The number of clock cycles to skip ranges from 0 (no clock skipping; 32/32 of the QCPU PLL frequency) to 23 (9/32 of the QCPU PLL frequency).

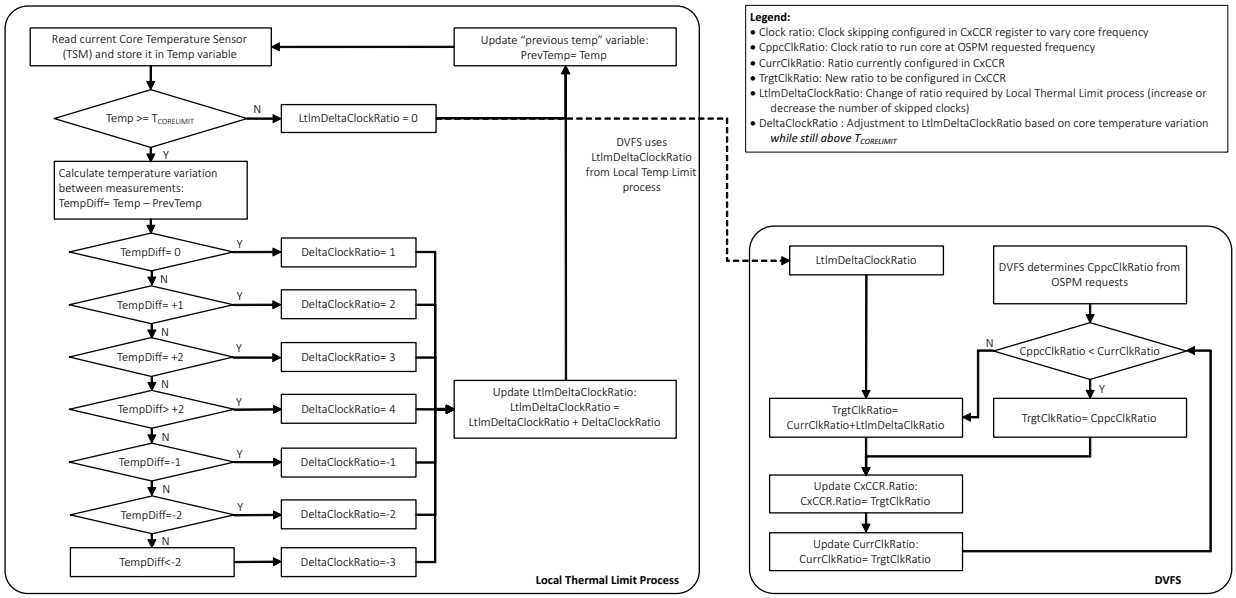
See *Figure 11-8* for a flowchart describing how DVFS uses these parameters to determine clock ratios.

11.6.1.3.2 Local Thermal Limit Management Algorithm

The local thermal limit management algorithm does not modify core clock frequency but instead provides DVFS with the adjustment required to reduce core frequency and bring it $T_{CORELIMIT}$.

Figure 11-8 illustrates how the local thermal limit management algorithm determines the clock skipping required to reduce core temperature.

Figure 11-8: Local Thermal Limit Management Algorithm and DVFS



The algorithm determines the value of the LtlmDeltaClockRatio variable to provide DVFS with additional required clock skipping, if any.

If the core temperature is below $T_{CORELIMIT}$, no adjustment is required and LtlmDeltaClockRatio = 0.

If the core temperature exceeds $T_{CORELIMIT}$, LtlmDeltaClockRatio is non-zero and its value is adjusted, depending upon the temperature difference between two measurements.

Here are some examples:

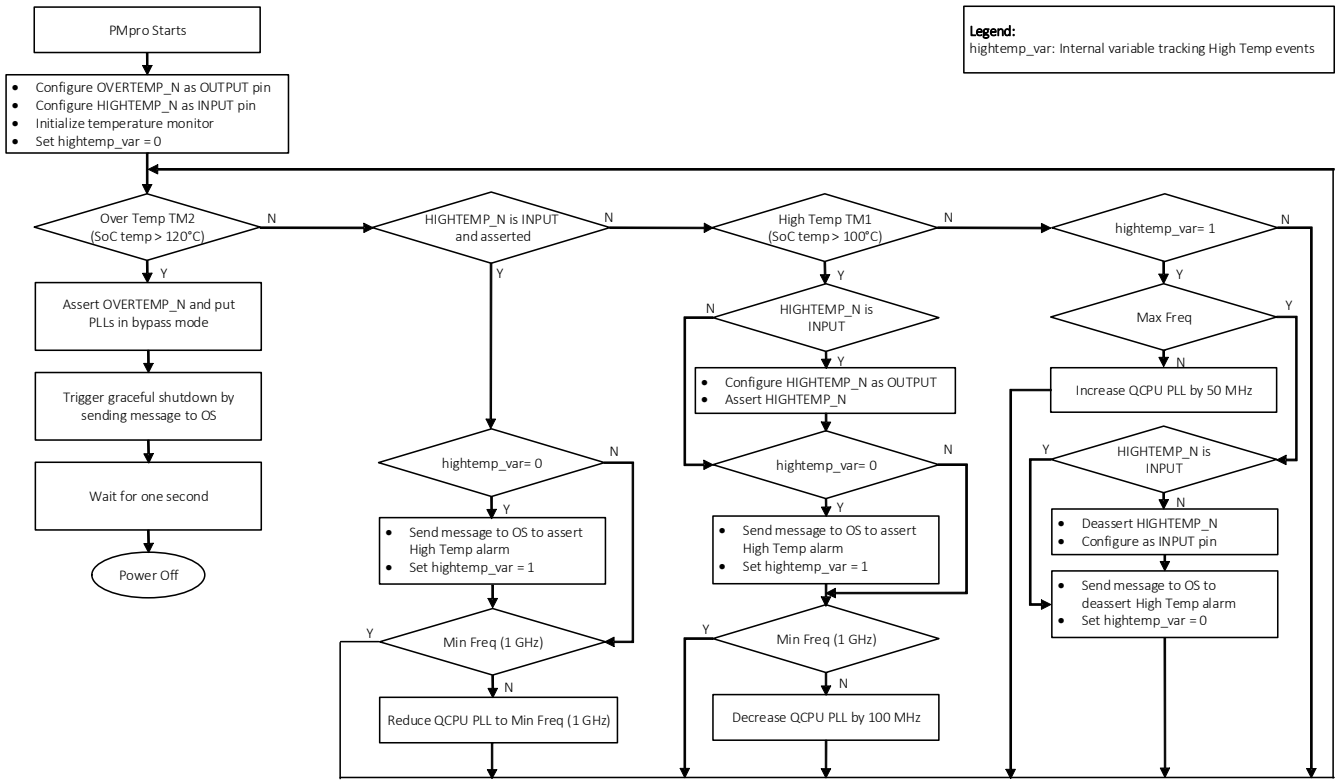
- If the core temperature measurements are the same and exceed $T_{CORELIMIT}$, LtlmDeltaClockRatio is increased by one cycle (DeltaClockRatio = 1) and DVFS increases clock skipping by one cycle, decreasing core frequency by 1/32 in the supported clock skipping range.
- If the core temperature increases by 2°C and still exceeds $T_{CORELIMIT}$, LtlmDeltaClockRatio is increased by three cycles (DeltaClockRatio = 3) and DVFS increases clock skipping by three cycles, decreasing core frequency by 3/32 in the supported clock skipping range)
- If core temperature decreases by 2°C but still exceeds $T_{CORELIMIT}$, LtlmDeltaClockRatio is decreased by two cycles (DeltaClockRatio =-2) and DVFS decreases clock skipping by two cycles, increasing core frequency by 2/32 in the supported clock skipping range.

11.6.1.4 Global Thermal Limits Management

Although local thermal limit management focuses on the thermal behavior of individual cores (and only cores), global thermal limits management focuses on the entire Altra Max processor.

Subsequent sections describe how processor temperatures are monitored and managed.

Figure 11-9: Global Thermal Limits Management Algorithm



11.6.1.4.1 Thermal Monitor (TM1)

Some sections of the die may exceed T_{cj} from time to time, based on activity. However, it is important to keep the die temperature from crossing the $T_{HIGHTEMP}$ threshold so that the Altra Max processor does not overheat. If the die temperature exceeds the $T_{HIGHTEMP}$ limit because of environmental conditions, such as a failed fan, or because of sustained high activity, such as a power-virus test, TM1 reduces the output frequency of the QCPU PLLs in an attempt to lower power dissipation.

Note: TM1 operation takes precedence over ACPI P-states, that is, over the DVFS processing of OSPM requests described in *Section 11.5, "Performance Feedback Counters."* Reducing the QCPU PLL clocks slows the frequency of the CPU cores, regardless of the P-State requested by ACPI. Because the PLL frequency changes, TM0 Changes the frequency defined for the P-States.

An ACPI P-State transition request to a frequency lower than the TM1 activation frequency is granted and changes the frequency to the requested P-State value, but only if TM1 is activated internally. However, an ACPI P-State transition request to a frequency higher than the TM1 activation frequency is not granted, and the request does not change the frequency. Additionally, external TM1 activation maintains QCPU PLLs at the minimum value of 1 GHz, and ACPI P-State requests are not granted.

TM0 Can be activated internally, by thermal sensors reporting temperatures above $T_{HIGHTEMP}$, or externally, by asserting the HIGHTEMP_N IO signal as an input. By default, HIGHTEMP_N is configured as an input.

11.6.1.4.1.1 Internal Activation

If TM1 is not already activated externally, PMpro can activate TM1 when $T_{HIGHSENSOR}$ equals or exceeds $T_{HIGHTEMP}$. The TM0 Controller reduces the QCPU PLL frequency to attempt to keep $T_{HIGHSENSOR}$ at or below $T_{HIGHTEMP}$. It is possible that frequency reduction cannot reduce the temperature below $T_{HIGHTEMP}$. If the temperature continues to rise above $T_{HIGHTEMP}$ while TM1 is activated, the temperature eventually may rise to $T_{OVERTEMP}$; at this point, Thermal Monitor 2 (TM2) is activated.

The state in which TM1 reduces the QCPU PLL from current values is called "active." While TM1 is active, the HIGHTEMP_N IO is configured as an output and driven active (pulled low). This tells system agents connected to this signal that the CPU clocks are throttled.

When the active state is exited, the HIGHTEMP_N signal is deasserted and configured as an input. The CPU clocks are restored to the Max Frequency as determined by DVFS. When TM1 is activated internally, it is not possible to detect that HIGHTEMP_N is asserted.

11.6.1.4.1.2 External Activation

If TM1 is not already activated internally, an external system agent can use the HIGHTEMP_N signal to activate TM1. This open-drain signal can be wire-ORed to other external signals. However, TM1 behaves differently when HIGHTEMP_N is asserted externally than when TM1 is activated internally.

When TM1 is not activated internally and HIGHTEMP_N is asserted (low), the QCPU PLL clock frequency is immediately reduced to its minimum 1 GHz frequency. When HIGHTEMP_N is deasserted, the CPM clock frequency is restored to its normal operating frequency.

11.6.1.4.1.3 Implementation

Internal TM1 monitoring is implemented using on-die thermal sensors and firmware running on PMpro. The firmware monitors the thermal sensors. If the corrected value of any sensor is equals or exceeds ($T_{HIGHTEMP}$), TM1 is activated, reducing the clock frequency of all QCPU PLLs in 50 MHz increments. The actual core frequency is reduced by a ratio of 50 MHz, as configured in the corresponding CxCCR register of each core.

Note: The minimum QCPU PLL frequency is 1 GHz, and a TM1 event does not reduce it below this minimum.

An N sample average of each thermal sensor in used as the value for that sensor. As long as the current high-temperature reading is greater than or equal to the previous reading, the clock frequency is reduced. If the current high temperature reading is less than the previous reading, the QCPU PLL remains the same.

When the temperature falls below $T_{HIGHTEMP}$ (in cases of internal TM1 activation), or when the HIGHTEMP_N input is deasserted (in cases of external activation), TM1 is deactivated and the QCPU PLL frequency is restored to its Max Frequency, as determined by DVFS, in 50 MHz increments.

11.6.1.4.2 Thermal Monitor (TM2)

TM2 is activated when $T_{HIGHSENSOR}$ equals or exceeds $T_{OVERTEMP}$, the temperature at which permanent damage can occur to the processor.

When TM2 is activated, OVERTEMP_N is asserted (low) and PLLs are put in bypass mode for low-frequency mode. PMpro then requests the OS to start a graceful shutdown and waits one second before sending a Power-Off request to the BMC.

11.6.1.5 Hardware Interfaces

These external interfaces relate to thermal management:

- Primary I^2C device interface.
  This interface controls and monitors the external VRMs.
- PMALERT_N.
  This open-drain signal implements the PMBus signal, PMALERT#, which indicates multipurpose alerts.
- HIGHTEMP_N.
  This open drain IO signal indicates that the maximum operating temperature is reached; equivalent to PROCHOT#.
- OVERTEMP_N.
  This output signal indicates that a critical temperature is reached; equivalent to THERMTRIP#.

11.6.1.5.1 Primary I^2C Device Interface

External VRMs provide a PMBus interface that supports access to all CSRs. The PMBus protocols run on the SMBus physical interface. The SMBus is a subset of the I^2C interface, which enables control of the VRMs using the primary I^2C device interface on the Altra Max processor. The PMBus specifies an additional signal, PMALERT#, which indicates various alerts to the primary I^2C device.

11.6.1.5.2 HIGHTEMP_N

The HIGHTEMP_N input signal, when asserted, activates TM1. When TM1 is activated internally, HIGHTEMP_N is enabled as an output and asserted (pulled low).

11.6.1.5.3 OVERTEMP_N

The OVERTEMP_N output signal is asserted when TM2 is activated (pulled low). This informs the system components when the Altra Max processor reaches $T_{OVERTEMP}$ and that voltage should be removed.

11.6.2 Power Limits

A set of events is identified in each PEM to represent consumed power. For each event, an independent counter simply counts the number of times the event occurs.

11.6.2.1 Power Estimates

To support limit management software, Altra Max processors provide chip power estimates for the CPUs, MCUs, and various blocks in the SoC power domain. During processor operation, PMpro computes static and dynamic power estimates.

11.6.2.1.1 Static Power

At test time, the leakage power of the part is measured and, based on previous characterization, parameters for estimating leakage power are fused into the part. Based on these parameters and a characterized leakage equation, PMpro continually estimates leakage power based on the current voltage and die temperature.

11.6.2.1.2 Dynamic Power

Dynamic power is constantly estimated and time-averaged to a thermal time scale, based on periodic sampling of arrays of power-event counters in all key processor blocks. The power-event counters measure logic and memory access activity that correlates with dynamic power.

PMpro computes instantaneous estimated dynamic power using these activity values, combined with a set of power equation coefficients determined by previous characterization. Overall Altra Max processor dynamic power is determined as follows:

- PCP domain dynamic power is estimated using a set of power event counters in each CPM and the CMI. A weighted sum of counter values for each agent provides an accurate estimate of its current dynamic power.
- SoC-domain dynamic power is estimated using a set of power-event counters in each MCU, and based on enabled and active IO interfaces. In practice, SoC domain power is a small fraction of overall power when the Altra Max processor operates at high power levels.
- DDR IO dynamic power is estimated using MCU power event counters.

When generated power increases, junction temperature typically also rises, but only after some delay. Sustained rising power can provide an early indication of when maximum/nominal frequency operation may need to be gradually reduced.

11.6.2.1.3 Power Event Monitors (PEMs)

An Altra Max processor has about 280 distributed PEMs, with two PEMs per core and other PEMs for other Altra Max processor components. PMpro initially programs the PEMs and uses measurements to continuously calculate a power estimate or activity factor.

11.6.2.1.4 Power Limits Management Algorithm

A software closed Proportional-Integral-Derivative (PID) control loop is implemented for power limits management. These parameters control the power limits management algorithm:

- Maximum TDP.
- PMIN input.
- BMC power limit.

PMpro periodically reads PEM weighted average data, calculates average power, and compares the new computed average power with the maximum TDP or BMC limit. The power limit management loop calculates the Power Limit Frequency Cap value.

11.6.2.1.5 PMIN

This input signal immediately throttles the cores and PCP to the lowest frequencies and voltage.

11.6.2.1.6 BMC Power Limit

The platform BMC can write to the dedicated "BMC register 0xE5" in SMpro to set a new power limit in an Altra Max processor. The BMC accesses this register through I^2C.

Refer to *Altra Max BMC Interface Specification* for more information about communication between BMC and the Altra Max processor, including "BMC Register 0xE5".

11.6.2.2 Core Max-Power Throttling

Each core includes a max-power throttling mechanism which reduces the average power consumed by high-power events in the Load-Store and Vector Execute units. It does this by throttling high-power events at a rate specified by the value of the MXP_TP CPUECTLR_EL1 Field, after the average rate of those high-power events exceeds the threshold specified by the value of the MXP_ATHR CPUECTLR_EL1 Field by 5%. Throttling continues until the average rate of high-activity events drops to or falls below the threshold specified by MXP_ATHR.

Arm recommends setting MXP_ATHR and MXP_TP as defined in *Table 11-12* to balance the effectiveness and responsiveness of the max-power throttling mechanism, while ensuring minimal impact to achieved performance caused by activity throttling.

Table 11-12: Core Max-Power Configuration

MXP_ATHR	MXP_TP	DESCRIPTION
70%	30 – 40%	Arm recommends using MXP_ATHR/MXP_TP values of 60%/40% or 70%/30%, depending upon how much average power reduction is required.
60%	30 – 40%	
50%	30 – 50%	Setting MXP_ATHR to the value of either 50% or 40% can lead to measurable performance degradation (1% – 2% for MXP_ATHR at 50%, 3% – 5% for MXP_ATHR at 40%) in workloads having a high density of high-power events.
40%	40 – 60%	

The max-power throttling mechanism is controlled by the CPUECTLR_EL1 Fields described in *Table 11-13*.

Table 11-13: CPUECTLR_EL0 Control for Max-Power Throttling

BIT	FIELD	DESCRIPTION
61	MXP_EN	Max-power throttle enable. 0: Disables max-power throttling mechanism. This is the reset value. 1: Enables max-power throttling mechanism.
58:57	MXP_TP	Percentage of throttling in the Load-Store and Vector Execute units during the period when throttling has been triggered and is active. This value dictates the rate at which the max-power throttling mechanism reduces average power, with higher values yielding faster reduction in average power, and lower values yielding slower reduction in average power. 00: Throttle by 60%. This is the reset value. 01: Throttle by 50%. 10: Throttle by 40%. 11: Throttle by 30%. The time required for throttling to effectively reduce average power ranges from 128 to 512 cycles.
56:55	MXP_ATHR	Peak activity threshold at which max-power throttling is triggered. 00: Max-power throttling triggered at 70% of peak activity. This is the reset value. 01: Max-power throttling triggered at 60% of peak activity. 10: Max-power throttling triggered at 50% of peak activity. 11: Max-power throttling triggered at 40% of peak activity.

11.6.3 Current Limits

In Altra Max processors, the current limit management (CLM) feature modulates CPU PLL frequencies to prevent current excursions above a specified current consumption threshold. The current consumption threshold for Altra Max processors depends upon SKU definitions and PCP VRM overcurrent (OC) warning threshold settings.

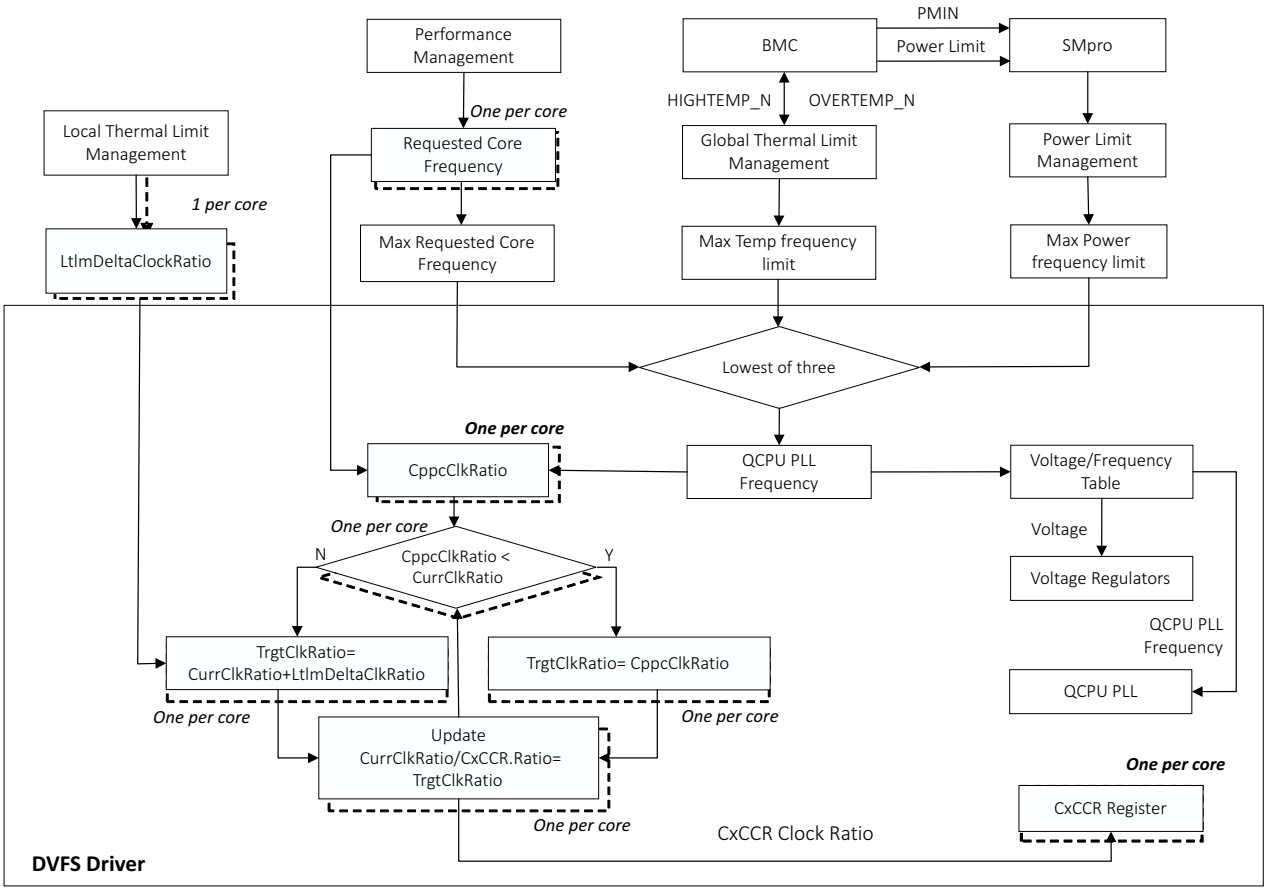
The CLM feature is configured using the following NVPARAMs:

- NV_SI_ALTRAMAX_ICCMAX_EN:
  Enable/disable the CLM feature.
- NV_SI_ALTRAMAX_CLM_CURR_MARGIN:
  PCP current margin for OC Limit, used to adjust the PCP VRM OC warning threshold for CLM.

11.7 Power Management Flow

Figure 11-10 illustrates how the performance management, power limit, and thermal limit flows merge as part of the DVFS driver to control processor frequencies.

Figure 11-10: Power Management Flow: Performance and Thermal and Power Limits



Generic Interrupt Controller (GIC)

12

This chapter describes the GIC and its use in an Altra Max processor. The chapter covers these topics:

12.1 Overview

The Altra Max GIC collects Interrupt Requests (IRQs) from on-chip and off-chip sources and delivers the IRQs to one or all cores in the CPMs. The GIC is based on and complies with *ARM Generic Interrupt Controller Architecture Specification Version 3.0 and 4.0.*

12.2 Private Peripheral Interrupts (PPIs)

A PPI is an interrupt that is specific to a particular core. PPIs are typically used for peripherals that are tightly coupled to a particular core. Interrupts connected to PPI inputs associated with one core are sent only to that core. Activating a PPI on one core does not affect the same interrupt ID on another core. The settings for each PPI are also independent between cores.

Table 12-1: Altra Max Processor PPIs

INTERRUPT ID	DESCRIPTION
16	Core fault interrupt
17	Core error recovery interrupt
18	Snoop control/SAM fault interrupt
21	Statistical profiling interrupt
22	COMMIRQ; Dual Clock Compare (DCC) interrupt
23	Performance monitors Interrupt
24	Generic debug/CTI output trigger event CTIIRQ
25	GIC maintenance interrupt
26	Overflow interrupt from CNTHP
27	Overflow interrupt from CNTV
28	Overflow interrupt from CNTHV
29	Overflow interrupt from CNTPS
30	Overflow interrupt from CNTP
31	Snoop control/SAM error recovery interrupt

12.3 Shared Peripheral Interrupts (SPIs)

SPIs are typically used for peripherals that are not tightly coupled to a particular core. An SPI can be programmed to target either one particular core or any core. Activating an SPI on one core activates the SPI for all cores. That is, at most one core activates an SPI. The settings for each SPI are also shared among all cores

Note: In *Table 12-2*, missing Interrupt IDs, for example, 64 – 69, are not driven by hardware.

Table 12-2: Altra Max Processor SPIs (Sheet 1 of 7)

INTERRUPT ID	DESCRIPTION
32	SMpro – CPUS0 DB AVAIL/ACK
33	SMpro – CPUS1 DB AVAIL/ACK
34	SMpro – CPUS2 DB AVAIL/ACK
35	SMpro – CPUS3 DB AVAIL/ACK
36	SMpro – CPUS4 DB AVAIL/ACK
37	SMpro – CPUS5 DB AVAIL/ACK
38	SMpro – CPUS6 DB AVAIL/ACK
39	SMpro – CPUS7 DB AVAIL/ACK
40	SMpro – CPU0 DB AVAIL/ACK
41	SMpro – CPU1 DB AVAIL/ACK
42	SMpro – CPU2 DB AVAIL/ACK
43	SMpro – CPU3 DB AVAIL/ACK
44	SMpro – CPU4 DB AVAIL/ACK
45	SMpro – CPU5 DB AVAIL/ACK
46	SMpro – CPU6 DB AVAIL/ACK
47	SMpro – CPU7 DB AVAIL/ACK
48	PMpro – CPUS0 DB AVAIL/ACK
49	PMpro – CPUS1 DB AVAIL/ACK
50	PMpro – CPUS2 DB AVAIL/ACK
51	PMpro – CPUS3 DB AVAIL/ACK
52	PMpro – CPUS4 DB AVAIL/ACK
53	PMpro – CPUS5 DB AVAIL/ACK
54	PMpro – CPUS6 DB AVAIL/ACK
55	PMpro – CPUS7 DB AVAIL/ACK
56	PMpro – CPU0 DB AVAIL/ACK
57	PMpro – CPU1 DB AVAIL/ACK

Table 12-2: Altra Max Processor SPIs (Sheet 2 of 7)

INTERRUPT ID	DESCRIPTION
58	PMpro – CPU2 DB AVAIL/ACK
59	PMpro – CPU3 DB AVAIL/ACK
60	PMpro – CPU4 DB AVAIL/ACK
61	PMpro – CPU5 DB AVAIL/ACK
62	PMpro – CPU6 DB AVAIL/ACK
63	PMpro – CPU7 DB AVAIL/ACK
70	I2C0
72	External IRQ0
73	External IRQ1
74	External IRQ2
75	External IRQ3
76	External IRQ4
77	External IRQ5
78	External IRQ6
79	External IRQ7
84	External IRQ8
85	External IRQ9
86	External IRQ10
87	External IRQ11
88	System Wake-up Timer0
89	Timer1
90	Timer2
91	Timer3
92	WS0 Watchdog NS
93	WS1 Watchdog NS
94	WS0 Watchdog S
95	External IRQ12
96	SPI0
97	SPI1
98	UART0
99	UART1

Table 12-2: Altra Max Processor SPIs (Sheet 3 of 7)

INTERRUPT ID	DESCRIPTION
100	UART2
101	UART3
102	UART4
103	I2C2
104	I2C3
105	I2C4
106	I2C5
107	I2C6
108	I2C7
109	I2C8
110	I2C9
111	I2C10
112	INTREQPPU (CMI)
113	INTREQERRS (CMI)
114	INTREQFAULTS (CMI)
117	External IRQ13
118	External IRQ14
119	External IRQ15
120	MCU0_int
121	MCU1_int
122	MCU2_int
123	MCU3_int
124	MCU4_int
125	MCU5_int
126	MCU6_int
127	MCU7_int
128	RCa0Pcie0_INTa, RCa0Pcie1_INTa, RCa0Pcie2_INTa, RCa0Pcie3_INTa
129	RCa0Pcie0_INTb, RCa0Pcie1_INTb, RCa0Pcie2_INTb, RCa0Pcie3_INTb
130	RCa0Pcie0_INTc, RCa0Pcie1_INTc, RCa0Pcie2_INTc, RCa0Pcie3_INTc
131	RCa0Pcie0_INTd, RCa0Pcie1_INTd, RCa0Pcie2_INTd, RCa0Pcie3_INTd
132	RCa1Pcie0_INTa, RCa1Pcie1_INTa, RCa1Pcie2_INTa, RCa1Pcie3_INTa

Table 12-2: Altra Max Processor SPIs (Sheet 4 of 7)

INTERRUPT ID	DESCRIPTION
133	RCa1Pcie0_INTb, RCa1Pcie1_INTb, RCa1Pcie2_INTb, RCa1Pcie3_INTb
134	RCa1Pcie0_INTc, RCa1Pcie1_INTc, RCa1Pcie2_INTc, RCa1Pcie3_INTc
135	RCa1Pcie0_INTd, RCa1Pcie1_INTd, RCa1Pcie2_INTd, RCa1Pcie3_INTd
136	RCa2Pcie0_INTa, RCa2Pcie1_INTa, RCa2Pcie2_INTa, RCa2Pcie3_INTa
137	RCa2Pcie0_INTb, RCa2Pcie1_INTb, RCa2Pcie2_INTb, RCa2Pcie3_INTb
138	RCa2Pcie0_INTc, RCa2Pcie1_INTc, RCa2Pcie2_INTc, RCa2Pcie3_INTc
139	RCa2Pcie0_INTd, RCa2Pcie1_INTd, RCa2Pcie2_INTd, RCa2Pcie3_INTd
140	RCa3Pcie0_INTa, RCa3Pcie1_INTa, RCa3Pcie2_INTa, RCa3Pcie3_INTa
141	RCa3Pcie0_INTb, RCa3Pcie1_INTb, RCa3Pcie2_INTb, RCa3Pcie3_INTb
142	RCa3Pcie0_INTc, RCa3Pcie1_INTc, RCa3Pcie2_INTc, RCa3Pcie3_INTc
143	RCa3Pcie0_INTd, RCa3Pcie1_INTd, RCa3Pcie2_INTd, RCa3Pcie3_INTd
144	RCa4Pcie0_INTa, RCa4Pcie1_INTa, RCa4Pcie2_INTa, RCa4Pcie3_INTa
145	RCa4Pcie0_INTb, RCa4Pcie1_INTb, RCa4Pcie2_INTb, RCa4Pcie3_INTb
146	RCa4Pcie0_INTc, RCa4Pcie1_INTc, RCa4Pcie2_INTc, RCa4Pcie3_INTc
147	RCa4Pcie0_INTd, RCa4Pcie1_INTd, RCa4Pcie2_INTd, RCa4Pcie3_INTd
148	RCa5Pcie0_INTa, RCa5Pcie1_INTa, RCa5Pcie2_INTa, RCa5Pcie3_INTa
149	RCa5Pcie0_INTb, RCa5Pcie1_INTb, RCa5Pcie2_INTb, RCa5Pcie3_INTb
150	RCa5Pcie0_INTc, RCa5Pcie1_INTc, RCa5Pcie2_INTc, RCa5Pcie3_INTc
151	RCa5Pcie0_INTd, RCa5Pcie1_INTd, RCa5Pcie2_INTd, RCa5Pcie3_INTd
152	RCa6Pcie0_INTa, RCa6Pcie1_INTa, RCa6Pcie2_INTa, RCa6Pcie3_INTa
153	RCa6Pcie0_INTb, RCa6Pcie1_INTb, RCa6Pcie2_INTb, RCa6Pcie3_INTb
154	RCa6Pcie0_INTc, RCa6Pcie1_INTc, RCa6Pcie2_INTc, RCa6Pcie3_INTc
155	RCa6Pcie0_INTd, RCa6Pcie1_INTd, RCa6Pcie2_INTd, RCa6Pcie3_INTd
156	RCa7Pcie0_INTa, RCa7Pcie1_INTa, RCa7Pcie2_INTa, RCa7Pcie3_INTa
157	RCa7Pcie0_INTb, RCa7Pcie1_INTb, RCa7Pcie2_INTb, RCa7Pcie3_INTb
158	RCa7Pcie0_INTc, RCa7Pcie1_INTc, RCa7Pcie2_INTc, RCa7Pcie3_INTc
159	RCa7Pcie0_INTd, RCa7Pcie1_INTd, RCa7Pcie2_INTd, RCa7Pcie3_INTd
200	MCU0UErr
201	MCU1UErr
202	MCU2UErr
203	MCU3UErr

Table 12-2: Altra Max Processor SPIs (Sheet 5 of 7)

INTERRUPT ID	DESCRIPTION
204	MCU4UErr
205	MCU5UErr
206	MCU6UErr
207	MCU7UErr
208	Rca0_UERR
209	Rca1_UERR
210	Rca2_UERR
211	Rca3_UERR
212	Rca4_UERR
213	Rca5_UERR
214	Rca6_UERR
215	Rca7_UERR
216	PMpro_ERR
217	SMpro_ERR
218	OCM_ERR
219	CMI_ERR_NS (INTREQERRNS)
220	GIC_Err_Int
264	MCU0Fault
265	MCU1Fault
266	MCU2Fault
267	MCU3Fault
268	MCU4Fault
269	MCU5Fault
270	MCU6Fault
271	MCU7Fault
272	TCU0Fault
273	TCU1Fault
274	TCU2Fault
275	TCU3Fault
276	TCU4Fault
277	TCU5Fault

Table 12-2: Altra Max Processor SPIs (Sheet 6 of 7)

INTERRUPT ID	DESCRIPTION
278	TCU6Fault
279	TCU7Fault
280	RcA0_Fault
281	RcA1_Fault
282	RcA2_Fault
283	RcA3_Fault
284	RcA4_Fault
285	RcA5_Fault
286	RcA6_Fault
287	RcA7_Fault
288	PMpro_FAULT
289	SMpro_FAULT
290	OCM_FAULT
291	CMI_FAULT_NS (INTREQFAULTNS)
292	GIC_Fault_Int
293	RcA0 – 1_TBU0-5_ras_irpt
294	RcA2 – 3_TBU0-5_ras_irpt
295	RcA4 – 5_TBU0-5_ras_irpt
296	RcA6 – 7_TBU0-5_ras_irpt
302	CPM0 – 3_PMU
303	CPM4 – 7_PMU
304	CPM8-11_PMU
305	CPM12-15_PMU
306	CPM016-19_PMU
307	CPM20 – 23_PMU
308	CPM24 – 27_PMU
309	CPM28 – 31_PMU
310	CPM32 – 35_PMU
311	CPM36 – 39_PMU
312	MCU0 – 7_PMU
313	TCU0 – 7_PMU

Table 12-2: Altra Max Processor SPIs (Sheet 7 of 7)

INTERRUPT ID	DESCRIPTION
314	CMI_PMU
315	GIC_PMU
316	RCA0 – 1_TBU0-5_PMU_irpt
317	RCA2 – 3_TBU0-5_PMU_irpt
318	RCA4 – 5_TBU0-5_PMU_irpt
319	RCA6 – 7_TBU0-5_PMU_irpt
320	External IRQ16
321	External IRQ17
322	External IRQ18
323	External IRQ19
324	External IRQ20
325	External IRQ21
326	External IRQ22
327	External IRQ23
328	Rca0_Event
329	Rca1_Event
330	Rca2_Event
331	Rca3_Event
332	Rca4_Event
333	Rca5_Event
334	Rca6_Event
335	Rca7_Event
336	Rca0_Error_int
337	Rca1_Error_int
338	Rca2_Error_int
339	Rca3_Error_int
340	Rca4_Error_int
341	Rca5_Error_int
342	Rca6_Error_int
343	Rca7_Error_int

12.4 Software Generated Interrupts (SGIs)

SGIs are inter-processor interrupts, that is, interrupts generated from one core and sent to other cores. Activating an SGI on one core does not affect the same interrupt ID on another core. When an SGI is sent to all cores, it is handled independently on each core. The settings for each SGI are also independent between cores.

System registers in the generating core are used to generate SGIs. Legacy software can write the Software Generated Interrupt Register (GICD_SGIR). Sixteen independent SGIs, ID0 – ID15, are recorded separately for each target core.

12.5 Locality-Specific Peripheral Interrupts (LPIs)

LPIs are typically used for peripherals that produce message-based interrupts. An LPI targets only one core at a time. LPIs are generated when the peripheral writes to the ITS, which also contains registers to control LPI generation and maintenance.

12.6 Message Signaled Interrupts (MSIs)/Message Signaled Interrupts eXtended (MSI-X)

LPIs serve the PCIe MSI/MSI-X interrupt generation function for the GIC. Each PCIe RC generates the appropriate write transaction to a standardized GIC address referred to as GIC ITS or the GITS_TRANSLATOR address. Data write transactions must include the interrupt vector and the Device ID supplied to the GIC.

12.7 Programming Model

The GIC programming model comprises global registers and per-core registers. All cores can access the global registers. Per-core registers are accessed at the designated offset for the particular core.

To suppress interrupts, software can mask interrupt sources or set the current task priority to a value above the interrupt priority level of interrupts to be masked. Note that in both cases, the GIC receives and stores IRQs so that they can be delivered later.

12.7.1 Generic Interrupt Controller (GIC) Address Maps

This section contains address maps for the GIC components, and for the ITS.

12.7.1.1 Generic Interrupts Controller (GIC) Components Address Map

In *Table 12-3*, *n* is the core number (0–127) and *i* is one of the ITS instances (07) summarized in *Table 12-4*, which contains the GIC ITS address map.

Table 12-3: GIC Components Address Map

COMPONENT		SOCKET 0 ADDRESS	SOCKET 1 ADDRESS	DESCRIPTION	SIZE
GIC	GIC address range				12 MB
	GICD	0x1000 0000 0000	0x5000 0000 0000	GICD main page	64 KB
	GICT	0x1000 0000 0000	0x5000 0000 0000	GIC trace and debug	64 KB
	GICP	0x1000 0000 0000	0x5000 0000 0000	GIC PMU	64 KB
	GITS*i*	0x1000 0004 0000 + *i**0x20000	0x5000 0004 0000 + *i**0x20000	GITS*i* ITS address page	64 KB
				GITS*i*_trans ITS translation page	64 KB
	GICR*n*	0x1000 0x0004 0000 + *i**0x100000 + *n**0x20000	0x5000 0x0004 0000 + *i**0x100000 + *n**0x20000	GICR*n* (LPI)	64 KB
				GICR*n* (SGI)	64 KB
Reserved region					-
GIC RAS and PMU status registers	PCP address range				2 MB
	RAS	0x1000 00C0 0000	0x5000 00C0 0000	RAS registers	64 KB
	PMU	0x1000 00C0 0000	0x5000 00C0 0000	PMU interrupt registers	64 KB
Reserved region					-
GIC configuration, error, and status registers	GIC configuration and error registers				2 MB
	CSRs	0x1000 00E0 0000	0x5000 00E0 0000	GIC CSRs	64 KB
	ITS configuration	0x1000 00E0 0000	0x5000 00E0 0000	ITS CSRs	64 KB

12.7.1.2 Generic Interrupts Controller (GIC) Interrupt Translation Service (ITS) Address Map

Table 12-4: GIC ITS Address Map

RC	ITS NUMBER	ITS ID	ITSTAGADR	
			SOCKET 0	SOCKET 1
RcA0	ITS0	0x00	0x1000 0005 0040	0x5000 0005 0040
RcA1	ITS1	0x01	0x1000 0007 0040	0x5000 0007 0040
RcA2	ITS2	0x02	0x1000 0009 0040	0x5000 0009 0040
RcA3	ITS3	0x03	0x1000 000B 0040	0x5000 000B 0040
RcA4	ITS4	0x04	0x1000 000D 0040	0x5000 000D 0040
RcA5	ITS5	0x05	0x1000 000F 0040	0x5000 000F 0040
RcA6	ITS6	0x06	0x1000 0010 0040	0x5000 0010 0040
RcA7	ITS7	0x07	0x1000 0010 0040	0x5000 0010 0040

Debug and Trace

13

This chapter describes the Altra Max processor debug architecture, infrastructure, functionality, and security. The chapter covers these topics:

13.1 Overview

Altra Max processors include hardware and software to support the Arm external debug mode. These processors also include hardware trace blocks that support real-time Embedded Trace Macrocell (ETM) tracing of instruction streams running in the Altra Max cores and debug flow control unit (flit) tracing in the CMI.

The processor is designed to the *ARMv8 Architecture Reference Manual, Revision F*, *Arm CoreSight Architecture Specification v3.0*, and *Arm CoreSight Base System Architecture 1.0, Combination B.*

For more information, refer to *Arm Debug Interface Architecture Specification ADIv6.0* for the PCP (Armv8) DAP, and to *Arm Debug Interface Architecture Specification ADIv5.0 to ADIv5.2* for the SMpro and PMpro DAPs.

13.2 Debug Hardware Components

- SMpro.
  - DAP.
    - JTAG Debug Port (JTAG-DP).
    - AHB Access Port (AHB-AP).
  - 32-bit Arm Cortex-M3 microcontroller.
    - Debug.
    - Data Watchpoint and Trace (DWT).
    - Flash Patch and Breakpoint (FPB).
- PMpro
  - DAP.
    - JTAG-DP.
    - AHB-AP.
  - 32-bit Arm Cortex-M3 microcontroller.
    - Debug.
    - DWT.
    - FPB.
- PCP (Armv8).
  - DAP.
    - JTAG-DP.
    - APB Interconnect (APBIC).
    - Primary ROM Table.
    - APB-AP.
    - AXI Access Port (AXI-AP).
    - SoC CTI.

- CPM.
  - CPM debug/trace (duplicated in each CPM).
    - TMC/ETR.
    - CATU.
    - CTI.
    - APB.
    - ATB.
    - AXI.
  - Core (duplicated in each core).
    - Debug.
    - CTI.
    - PMU.
    - ETM.
    - Activity Monitor Unit (AMU).
    - APB.
    - ATB.
- CMI Trace.
  - DTMs (duplicated for each crosspoint).
  - DTC (duplicated for each CMI slice).
  - TMC/ETF/Embedded Trace Buffer (ETB).
  - TMC/ETR.
  - CATU.
  - CTI.
  - APB.
  - ATB.
  - AXI.
- CTM.

13.3 Debug Subsystem Diagrams

Figure 13-1 illustrates the major debug subsystem components and their relationships.

Figure 13-1: Debug Subsystem Block Diagram

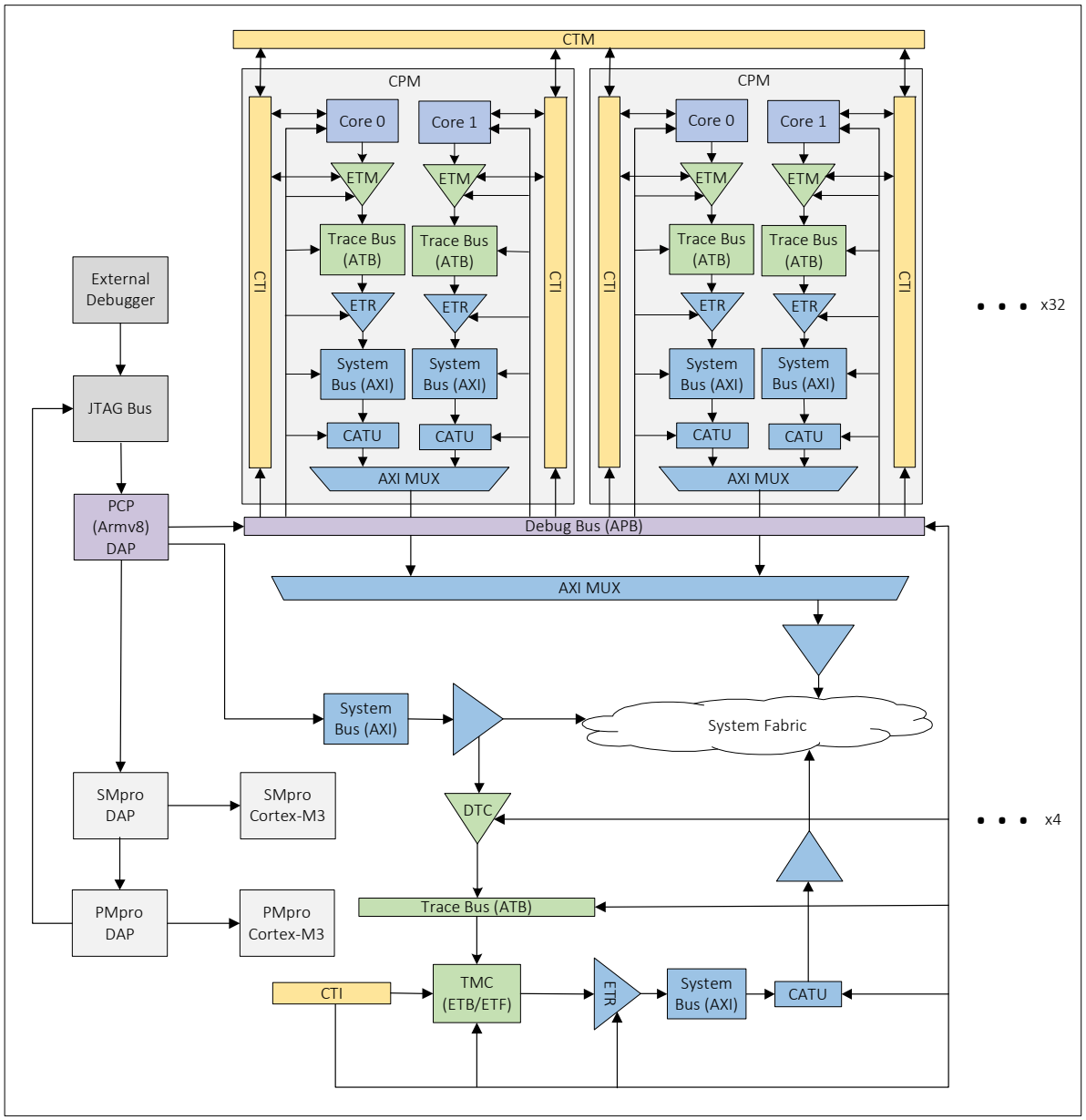
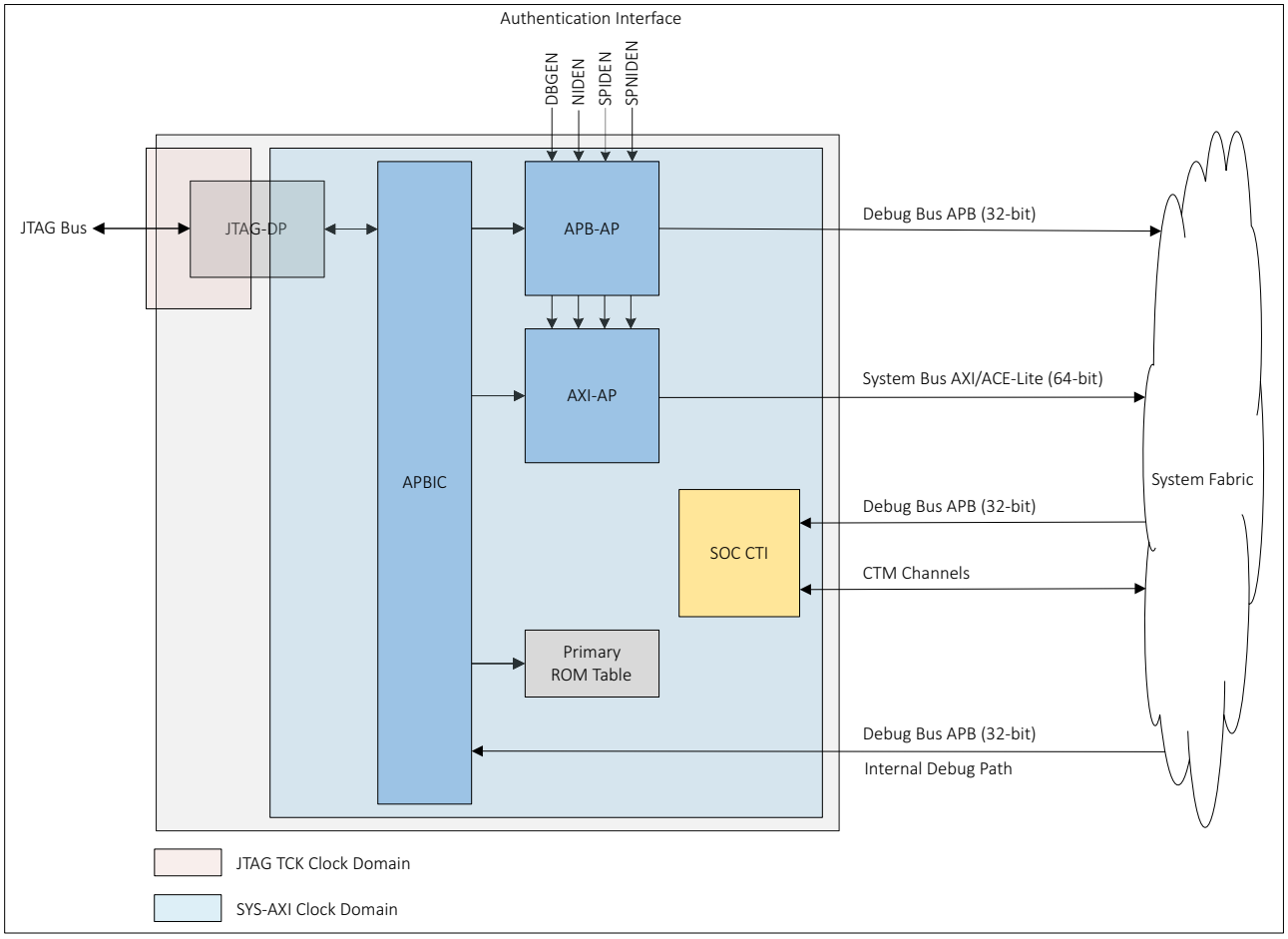


Figure 13-2 provides a more detailed view of the PCP (Armv8) DAP components.

Figure 13-2: PCP (Armv8) DAP Block Diagram



13.4 Joint Test Action Group (JTAG) Debug Interfaces

An Altra Max processor provides four JTAG debug interfaces in two categories:

- SoC debug/test:
  - SoC Test Access Port (TAP) – used for manufacturing and RMA testing.
- System debug:
  - PCP (Armv8) DAP – used to debug the PCP.
  - SMpro DAP – used to debug the SMpro ROM and firmware.
  - PMpro DAP – used to debug PMpro firmware.

13.5 Debug Security

Altra Max processors provide useful features that support processor security while debugging.

13.5.1 Debug Signals

Each debug domain provides these debug signals:

- DBGEN – Invasive debug enable.
- NIDEN – Non-invasive debug enable.
  - General control of CPU and CMI trace and PMU capabilities.
- SPIDEN – Secure invasive debug enable.
  - Enables debugging while processor is in the EL3, SEL1, and SEL0 exception levels.
  - Enables dumping or viewing cache lines, events, and traffic.
- SPNIDEN – Secure non-invasive debug enable.
  - Limits Altra Max, CMI trace, and PMU capabilities to non-secure world traffic and events.

When debug is disabled out-of-reset, it is possible to re-enable debug through firmware at later boot stages using write-once debug control registers for non-secure debug control and secure debug control.

Because the debug signals have no relevance until the PCP is initialized and brought out of reset during later boot stages, the debug signals are always disabled at power-on reset and are controlled by firmware using the debug control registers.

SMpro firmware drives secure debug signals based on TMM eFuses if no valid debug certificate is present. Otherwise, the debug signals are set based on the DCU mask provided in the debug certificate.

ATF drives the non-secure debug signals based on TMM eFuses if no valid non-secure debug certificate is provided with the authenticated UEFI image.

13.5.2 Processor Life Cycle State (LCS) and Debug

Any system reset must first query the current LCS before determining whether the debug facilities are accessible.

- CM LCS: JTAG/debug access is fully enabled out-of-reset.
- Secure LCS: JTAG/debug access is fully disabled out-of-reset.
- RMA LCS: JTAG/debug access is fully enabled out-of-reset.

In the Secure LCS, JTAG and debug capabilities are disabled out of reset. In the Secure LCS, it is possible to re-enable SMpro and PMpro DAP using a write-once TMMSDBGCR register.The secure boot ROM processes debug control bits in the TMM eFuse array to determine which debug capabilities are to be enabled on boot, as described in *Section 13.5.3, "Debug Control Unit (DCU) eFuses."* It is possible to re-enable debug capabilities that are fused out using an appropriate debug certificate, which is typically signed to enable debug for a specific system. For example, SMpro firmware re-enables debugging only when a valid certificate with the appropriate DCU mask (in certificate) is provided.

The SoC TAP can be re-enabled only by putting the part into the RMA LCS.

13.5.3 Debug Control Unit (DCU) eFuses

Customers decide what debugging and security trade-offs are acceptable for production parts, and are responsible for blowing the appropriate TMM eFuses at appropriate project phases, such as full debug during the Engineering Validation Test (EVT) phase, and no secure debug during the Design Verification Test (DVT) phase.

Special debug certificates are provided to customers to re-enable fine-grained debug control as needed.

13.6 Debug Address Maps

13.6.1 Altra Max (Armv8) DAP Address Map

Table 13-1 contains the Altra Max processor (Armv8) DAP address map, accessible to an external debugger using the DAP JTAG Debug Port (JTAG-DP).

Table 13-1: Altra Max (Armv8) DAP Address Map

START ADDRESS	END ADDRESS	SIZE	ASSIGNMENT
0x0000 0000	0x0000 0FFF	4 KB	Primary ROM Table
0x0001 0000	0x0001 0FFF	4 KB	APB-AP (External debug)
0x0002 0000	0x0002 0FFF	4 KB	AXI-AP

13.6.2 Debug Address Map

Table 13-2 contains the debug address map, accessible to an external debugger or internal debugger using the DAP APB-AP.

Table 13-2: External and Self-Hosted Debug Address Map (Sheet 1 of 4)

DEBUG APB ADDRESS RANGE		SIZE	ASSIGNMENT
START ADDRESS	END ADDRESS		
0x0004 0000	0x0004 0FFF	4 KB	Secondary ROM Table
0x0004 1000	0x0006 FFFF	–	Reserved
0x0007 0000	0x0007 0FFF	4 KB	SoC CTI
0x0007 1000	0x9FFF FFFF	–	Reserved
0xA000 0000	0xA03F FFFF	4 MB	CPM 0
0xA040 0000	0xA07F FFFF	4 MB	CPM 1
0xA080 0000	0xA0BF FFFF	4 MB	CPM 2
0xA0C0 0000	0xA0FF FFFF	4 MB	CPM 3
0xA100 0000	0xA13F FFFF	4 MB	CPM 4

Table 13-2: External and Self-Hosted Debug Address Map (Sheet 2 of 4)

DEBUG APB ADDRESS RANGE		SIZE	ASSIGNMENT
START ADDRESS	END ADDRESS		
0xA140 0000	0xA17F FFFF	4 MB	CPM 5
0xA180 0000	0xA1BF FFFF	4 MB	CPM 6
0xA1C0 0000	0xA1FF FFFF	4 MB	CPM 7
0xA200 0000	0xA23F FFFF	4 MB	CPM 8
0xA240 0000	0xA27F FFFF	4 MB	CPM 9
0xA280 0000	0xA2BF FFFF	4 MB	CPM 10
0xA2C0 0000	0xA2FF FFFF	4 MB	CPM 11
0xA300 0000	0xA33F FFFF	4 MB	CPM 12
0xA340 0000	0xA37F FFFF	4 MB	CPM 13
0xA380 0000	0xA3BF FFFF	4 MB	CPM 14
0xA3C0 0000	0xA3FF FFFF	4 MB	CPM 15
0xA400 0000	0xA43F FFFF	4 MB	CPM 16
0xA440 0000	0xA47F FFFF	4 MB	CPM 17
0xA480 0000	0xA4BF FFFF	4 MB	CPM 18
0xA4C0 0000	0xA4FF FFFF	4 MB	CPM 19
0xA500 0000	0xA53F FFFF	4 MB	CPM 20
0xA540 0000	0xA57F FFFF	4 MB	CPM 21
0xA580 0000	0xA5BF FFFF	4 MB	CPM 22
0xA5C0 0000	0xA5FF FFFF	4 MB	CPM 23
0xA600 0000	0xA63F FFFF	4 MB	CPM 24
0xA640 0000	0xA67F FFFF	4 MB	CPM 25
0xA680 0000	0xA6BF FFFF	4 MB	CPM 26
0xA6C0 0000	0xA6FF FFFF	4 MB	CPM 27
0xA700 0000	0xA73F FFFF	4 MB	CPM 28
0xA740 0000	0xA77F FFFF	4 MB	CPM 29
0xA780 0000	0xA7BF FFFF	4 MB	CPM 30
0xA7C0 0000	0xA7FF FFFF	4 MB	CPM 31
0xA800 0000	0xA83F FFFF	4 MB	CPM 32
0xA840 0000	0xA87F FFFF	4 MB	CPM 33
0xA880 0000	0xA8BF FFFF	4 MB	CPM 34

Table 13-2: External and Self-Hosted Debug Address Map (Sheet 3 of 4)

DEBUG APB ADDRESS RANGE		SIZE	ASSIGNMENT
START ADDRESS	END ADDRESS		
0xA8C0 0000	0xA8FF FFFF	4 MB	CPM 35
0xA900 0000	0xA93F FFFF	4 MB	CPM 36
0xA940 0000	0xA97F FFFF	4 MB	CPM 37
0xA980 0000	0xA9BF FFFF	4 MB	CPM 38
0xA9C0 0000	0xA9FF FFFF	4 MB	CPM 39
0xAA00 0000	0xAA3F FFFF	4 MB	CPM 40
0xAA40 0000	0xAA7F FFFF	4 MB	CPM 41
0xAA80 0000	0xAABF FFFF	4 MB	CPM 42
0xAAC0 0000	0xAAFF FFFF	4 MB	CPM 43
0xAB00 0000	0xAB3F FFFF	4 MB	CPM 44
0xAB40 0000	0xAB7F FFFF	4 MB	CPM 45
0xAB80 0000	0xABBF FFFF	4 MB	CPM 46
0xABC0 0000	0xABFF FFFF	4 MB	CPM 47
0xAC00 0000	0xAC3F FFFF	4 MB	CPM 48
0xAC40 0000	0xAC7F FFFF	4 MB	CPM 49
0xAC80 0000	0xACBF FFFF	4 MB	CPM 50
0xACC0 0000	0xACFF FFFF	4 MB	CPM 51
0xAD00 0000	0xAD3F FFFF	4 MB	CPM 52
0xAD40 0000	0xAD7F FFFF	4 MB	CPM 53
0xAD80 0000	0xADBF FFFF	4 MB	CPM 54
0xADC0 0000	0xADFF FFFF	4 MB	CPM 55
0xAE00 0000	0xAE3F FFFF	4 MB	CPM 56
0xAE40 0000	0xAE7F FFFF	4 MB	CPM 57
0xAE80 0000	0xAEBF FFFF	4 MB	CPM 58
0xAEC0 0000	0xAEFF FFFF	4 MB	CPM 59
0xAF00 0000	0xAF3F FFFF	4 MB	CPM 60
0xAF40 0000	0xAF7F FFFF	4 MB	CPM 61
0xAF80 0000	0xAFBF FFFF	4 MB	CPM 62
0xAFC0 0000	0xAFFF FFFF	4 MB	CPM 63
0xB000 0000	0xB4FF FFFF	–	Reserved

Table 13-2: External and Self-Hosted Debug Address Map (Sheet 4 of 4)

DEBUG APB ADDRESS RANGE		SIZE	ASSIGNMENT
START ADDRESS	END ADDRESS		
0xB600 0000	0xB63F FFFF	4 MB	CMI 0 (slice 0)
0xB640 0000	0xB67F FFFF	4 MB	CMI 1 (slice 1)
0xB680 0000	0xB6BF FFFF	4 MB	CMI 2 (slice 2)
0xB6C0 0000	0xB6FF FFFF	4 MB	CMI 3 (slice 3)

13.6.3 Cluster Processor Module (CPM) Debug Map

See *Table 13-2* for CPM 0–CPM 63 addresses.

The 64 KB page offsets listed in *Table 13-3* are for the CPM debug components in each CPM.

Table 13-3: CPM Debug Map

OFFSET	SIZE	COMPONENT
0x0000 0000	64 KB	CPM ROM Table
0x0001 0000	64 KB	Core 0 ETR
0x0002 0000	64 KB	Core 0 CATU
0x0001 0000	64 KB	Core 1 ETR
0x0002 0000	64 KB	Core 1 CATU
0x0005 0000	64 KB	CPM CTI

13.6.4 Core 0 and Core 1 Debug Map

The 64 KB page offsets listed in *Table 13-4* are for the debug components for the cores in each CPM. The CPM addresses are in *Table 13-2*.

Table 13-4: Core 0 and Core 1 Debug Map

OFFSET	SIZE	COMPONENT
0x0021 0000	64 KB	Core 0 Debug
0x0022 0000	64 KB	Core 0 CTI
0x0023 0000	64 KB	Core 0 PMU
0x0024 0000	64 KB	Core 0 ETM
0x0001 0000	64 KB	Core 0 ETR
0x0002 00000	64 KB	Core 0 CATU

Table 13-4: Core 0 and Core 1 Debug Map

OFFSET	SIZE	COMPONENT
0x0031 0000	64 KB	Core 1 Debug
0x0032 0000	64 KB	Core 1 CTI
0x0033 0000	64 KB	Core 1 PMU
0x0034 0000	64 KB	Core 1 ETM
0x0003 0000	64 KB	Core 1 ETR
0x0004 00000	64 KB	Core 1 CATU

13.6.5 Debug Trace Controller (DTC) Debug Map

The 64 KB page offsets listed in *Table 13-5* are for the debug components in each DTC.

Table 13-5: DTC Debug Map

OFFSET	SIZE	COMPONENT
0x0000 0000	64 KB	DTC ROM Table
0x0001 0000	64 KB	DTC ETF
0x0002 0000	64 KB	DTC ETR
0x0003 0000	64 KB	DTC CATU
0x0004 0000	64 KB	DTC CTI

13.6.6 Internal Debug APB-AP Address Map

Table 13-6 contains the Internal Debug APB-AP address map.

Table 13-6: Internal Debug APB-AP Address Map

SOCKET	START ADDRESS	END ADDRESS	SIZE	ASSIGNMENT
Socket 0	0x1000 C001 1000	0x1000 C001 1FFF	4 KB	APB-AP (Internal Debug)
Socket 1	0x5000 C001 1000	0x5000 C001 1FFF	4 KB	APB-AP (internal debug)

Glossary

A

Table A-1: Acronyms and Abbreviations (Sheet 1 of 11)

TERM	MEANING
1DPC	One DIMM per Channel
1P	Single-Socket Platform
2DPC	Two DIMMs per Channel
2P	Dual-Socket Platform
ACPI	Advanced Configuration and Power Interface
ACS	Access Control Services
AER	Advanced Error Reporting
AEST	Arm Error Source Table
AHB	Advanced High-Performance Bus
AHB-AP	AHB Access Port
AHBC	AHB Controller
ALI	Ampere Link Interconnect
AMBA	Advanced Microcontroller Bus Architecture
AMU	Activity Monitor Unit
ANC	Ampere NUMA Control
AP	Access Port
APB	Advanced Peripheral Bus
APBIC	APB Interconnect
APEI	ACPI Platform Error Interface
ARI	Alternative Routing ID, Alternative Routing Interpretation

Table A-1: Acronyms and Abbreviations (Sheet 2 of 11)

TERM	MEANING
ATB	AMBA Trace Bus
ATF	Arm Trusted Firmware
ATS	Address Translation Service
AVS	Adaptive Voltage Scaling
AXI	Advanced eXtensible Interface
AXI-AP	AXI Access Port
b	Bit
B	Byte
BERT	Boot Error Record Table
BIOS	Basic Input/Output System
BIST	Built-In Self Test
BL1	Boot Loader Stage 1
BL2	Boot Loader Stage 2
BMC	Baseboard Management Controller
BSC	Bootstrap Controller
CA	Command/Address
CAR	Clear After Read
CATU	CoreSight Address Translation Unit
CB	Configuration Bus
CCIX	Cache Coherent Interconnect for Accelerators
CCTI	CoreSight Cross-Trigger Interface
CDC	Clock Domain Crossing
CDM	Configuration-Dependent Module
CDR	Clock and Data Recovery
CE	Correctable Error
CFG	Configuration node
CFI	Corrected Error Interrupt
CHI	Coherent Hub Interface
CID	Chip ID
CMI	Coherent Mesh Interconnect
CoT	Chain of Trust

Table A-1: Acronyms and Abbreviations (Sheet 3 of 11)

TERM	MEANING
CPER	Common Platform Error Records
CPM	Cluster Processor Module
CPPC	Collaborative Processor Performance Control
CPU	One Altra Max core in a CPM; Processing Element (PE)
CRB	Command Response Buffer
CRC	Cyclic Redundancy Check
CS	Chip Select
CSR	Control and Status Register, Configuration and Status Register
CTI	Cross-Trigger Interface
CTM	Cross Trigger Matrix
CXS	CCIX Stream Interface
DAP	Debug Access Port
DB	Allow Database Key
DBID	Data Buffer ID
DCC	Dual Clock Compare
DCU	Debug Control Unit
DE	Deferred Error
DECERR	Decode Error
DED	Double Error Detecting
DES	Data Encryption Standard
DFS	Dynamic Frequency Scaling
DFT	Design For Test
DIP	Dynamic Insertion Policy
DLCMSM	Data Link Control and Management State Machine
DLLP	Data Link Layer Packet
DMA	Direct Memory Access
DMI	Direct Media Interface
DMT	Direct Memory Transfer
Doubleword	Eight bytes
DP	Debug Port
DPC	DIMMs Per Channel

Table A-1: Acronyms and Abbreviations (Sheet 4 of 11)

TERM	MEANING
DR	Data Register
DRAM	Dynamic Random Access Memory; Data RAM
DRAM	Dynamic RAM, Data RAM
DSDT	Differentiated System Description Table
DT	Debug and Trace
DTC	Debug and Trace Controller
DTI	Distributed Translation Interface
DTM	Debug and Trace Monitor
DVFS	Dynamic Voltage and Frequency Scaling
DVM	Distributed Virtual Memory
DVS	Dynamic Voltage Scaling
DVT	Design Verification Test
Dword	Eight bytes; Doubleword
DWT	Data Watchpoint and Trace
ECAM	Enhanced Configuration Access Mechanism
ECC	Error Correction Code
ED	Error Detection
EDC	Error Detection Code
EEPROM	Electrically Erasable Programmable Read-Only Memory
EINJ	Error Injection
EL	Exception Level
EP	Endpoint
ESM	Extended Speed Mode
ETB	Embedded Trace Buffer
ETM	Embedded Trace Macrocell
ETR	Embedded Trace Router
EVT	Engineering Verification Test
FCS	Frame Check Sequence
FI	Fault Interrupt
FIFO	First-In First-Out
FMADD	Fused Multiply-Add

Table A-1: Acronyms and Abbreviations (Sheet 5 of 11)

TERM	MEANING
FPB	Flash Patch and Breakpoint
GB	Gigabyte
Gb	Gigabit
GbE	Gigabit Ethernet
GCLK	Global Clock
GED	Generic Event Device
Gen	Generation
GHES	Generic Hardware Error Source
GIC	Generic Interrupt Controller
GPI	General-Purpose Input
GPIO	General-Purpose Input/Output
GT	Generic Timer
GT/s	Gigatransfers per second
GUID	Globally Unique ID
HA	Home Agent
Halfword	Two bytes; 16 bits
HBM	High Bandwidth Memory
HEST	Hardware Error Source Table
HUK	Hardware Unique Key
I^2C	Inter-Integrated Circuit
IANA	Internet Assigned Numbers Authority
ICV	Integrity Check Value
ID	Identifier
Int	Interrupt
IO	Input/Output
IRAM	Instruction RAM
IRQ	Interrupt Request
ITS	Interrupt Translation Service
JTAG	Joint Test Action Group
JTAG-DP	JTAG Debug Port
KB	Kilobyte

Table A-1: Acronyms and Abbreviations (Sheet 6 of 11)

TERM	MEANING
L1	Level 1
L2	Level 2
LCS	Life Cycle State
LPI	Locality-specific Peripheral Interrupt; Low-Power Interface; Lower Power Idle
LS	Low Speed
LSb	Least Significant Bit
LSB	Least Significant Byte
MB	Megabyte
MCU	Memory Controller Unit
MESI	Modified, Exclusive, Shared, Invalid
MMIO	Memory-Mapped IO
MMU	Memory Management Unit
MPS	Maximum Payload Size
MSI	Message Signaled Interrupt
MSI-X	Message Signaled Interrupt eXtended
MT/s	Megatransfers per second
n/a	not applicable
NA	Normalized Address; Not Applicable
NIST	National Institute of Standards and Technology
NTB	Non-Transparent Bridging
NUMA	Non-Uniform Memory Access
NVM	Nonvolatile Memory
NVMe	NVM Express
OCM	On-Chip Memory
ODM	Original Design Manufacturer
OEM	Original Equipment Manufacturer
OOB	Out of Band
OS	Operating System, Ordered Set
OSPM	Operating System-Directed Configuration and Power Management
OTP	One-Time Pad
P-State	Performance state

Table A-1: Acronyms and Abbreviations (Sheet 7 of 11)

TERM	MEANING
P2P	Peer-to-Peer
PA	Physical Address; Processing Agent
PASID	Process Address Space ID
PCC	Platform Communication Channel
PCIe	Peripheral Component Interconnect Express
PCP	Processor Complex
PE	Processing Element
PEM	Power Events Monitor
PHY	Physical Layer
PID	Proportional–Integral–Derivative
PKCS	Public-Key-Cryptography Standards
PLL	Phase-Locked Loop
PME	Power Management Event
PMpro	Power Management Processor
PMU	Performance Monitoring Unit
PoC	Point of Coherency
POR	Power On Reset
PoS	Point of Serialization
PPC	Processor Performance Control
PPI	Private Peripheral Interrupt
PRD	Physical Region Descriptor
PRI	Page Request Interface
PRNG	Pseudorandom Number Generator
Processor	An entire chip containing multiple cores, that is, multiple CPUs
PSCI	Power State Coordination Interface
PTM	Program Trace Macrocell
PVS	Process Voltage Scaling
QCPU	Quad CPU
QSPI	Quad Serial Peripheral Interface
Quadword	16 bytes
RA	Request Agent

Table A-1: Acronyms and Abbreviations (Sheet 8 of 11)

TERM	MEANING
RAM	Random Access Memory
RAS	Reliability, Availability, and Serviceability
RAZ	Read As Zero
RAZ/WI	Read As Zero, Writes Ignored
RC	Root Complex
RcA	RC Type A
RDIMM	Registered DIMM
REQ	Request (CHI channel)
RMA	Returned Merchandise Authorization
RMW	Read-Modify-Write
RO	Read-Only; Relaxed Ordering
ROM	Read-Only Memory
RoT	Root of Trust
RoTPK	Root of Trust Public Key
RP	Root Port
RSA	Rivest–Shamir–Adleman
Rsvd	Reserved
RW	Read/Write
RW1C	Read/Write-One-to-Clear
SAM	System Address Map
SAS	System Address Space
SBBR	Server Base Boot Requirements
SBSA	Server Base System Architecture
SC	System Counter; Self-Clear
SCI	System Control Interrupt
SEC	Single Error Correcting; Security
SECDED	Single Error Correction and Double Error Detection
SEL	System Event Log
SerDes	Serializer/Deserializer
SG-DMA	Scatter-Gather DMA
SGI	Software-Generated Interrupt

Table A-1: Acronyms and Abbreviations (Sheet 9 of 11)

TERM	MEANING
SGL	Scatter-Gather List
SHA	Secure Hash Algorithm
SII	System Input Interface; Switch Inbound Interface
SIMD	Single Instruction, Multiple Data
SIP	Static Insertion Policy
SKU	Stock-Keeping Unit
SLC	System Level Cache
SLVERR	Secondary Device Error
SMB, SMBus	System Management Bus
SMBIOS	System Management BIOS
SMC	Secure Monitor Call
SMMU	System Memory Management Unit
SMP	Symmetric Multiprocessing
SMpro	System Management Processor
SMT	Simultaneous Multithreading
SN-F	CHI Secondary Device Node
SNP	Snoop (CHI channel)
SoC, SOC	System-on-Chip
SOF	Start of frame
SOI	System Output Interface; Switch Outbound Interface
SP	Single Precision
SPD	Serial Presence Detect
SPI	Serial Peripheral Interface; Shared Peripheral Interrupt; Security Parameter Index
SR	Status Register; System Register
SR-IOV	Single-Root IO Virtualization
SRAT	Static Resource Affinity Table
SRC	Source
SRE	System Register Interface Enable
SRIS	Separate Refclk Independent SSC Architecture
SRNS	Separate Refclk No SSC Architecture
SRTM	Secure RoT Measurement

Table A-1: Acronyms and Abbreviations (Sheet 10 of 11)

TERM	MEANING
SS	Super Speed
SSC	Spread Spectrum Clocking
SSCG	Spread Spectrum Clock Generator
SSD	Solid-State Drive
SVE	Scalable Vector Extension
TAP	Test Access Port
TBU	Translation Buffer Unit
TC	Traffic Class
TCB	Test Control Block
TCG	Trusted Computing Group
TCU	Translation Control Unit
TDP	Thermal Design Power
TF	Trusted Firmware
TLP	Transaction Layer Packet, Thread-Level Parallelism
TMM	Trusted Management Module
TPH	TLP Processing Hint
TPIU	Trace Port Interface Unit
TPM	Trusted Platform Module
TQ	Transaction Queue
TRB	Transfer Request Block
TRM	Technical Reference Manual
TRNG	True Random Number Generator
TS1	Training Sequence 1
TS2	Training Sequence 2
TSM	Temperature Sensor Monitor
TSV	Through-Silicon Via
UART	Universal Asynchronous Receiver/Transmitter
UDIMM	Unregistered DIMM
UE	Uncorrectable Error
UEFI	Unified Extensible Firmware Interface
UI	Uncorrected Interrupt

Table A-1: Acronyms and Abbreviations (Sheet 11 of 11)

TERM	MEANING
ULM	Unit Level Module, Upper Level Module
UR	Unsupported Request
USB	Universal Serial Bus
VA	Virtual Address
VC	Virtual Channel
VCO	Voltage Controlled Oscillator
VDM	Vendor Defined Message, Voltage Droop Mitigation
VIPT	Virtually Indexed, Physically Tagged
VLAN	Virtual LAN
VM	Virtual Machine, Virtual Memory
VRM	Voltage Regulator Module
VRRP	Virtual Router Redundancy Protocol
WBQ	Write Back Queue
WDB	Write Data Buffer
WFE	Wait For Event
WFI	Wait For Interrupt
WI	Writes Ignored
WOCLR	Write One to Clear
Word	4 bytes; 32 bits
WRR	Weighted Round Robin
xHCI	eXtensible Host Controller Interface
XP	Crosspoint
XSDT	eXtended System Description Table