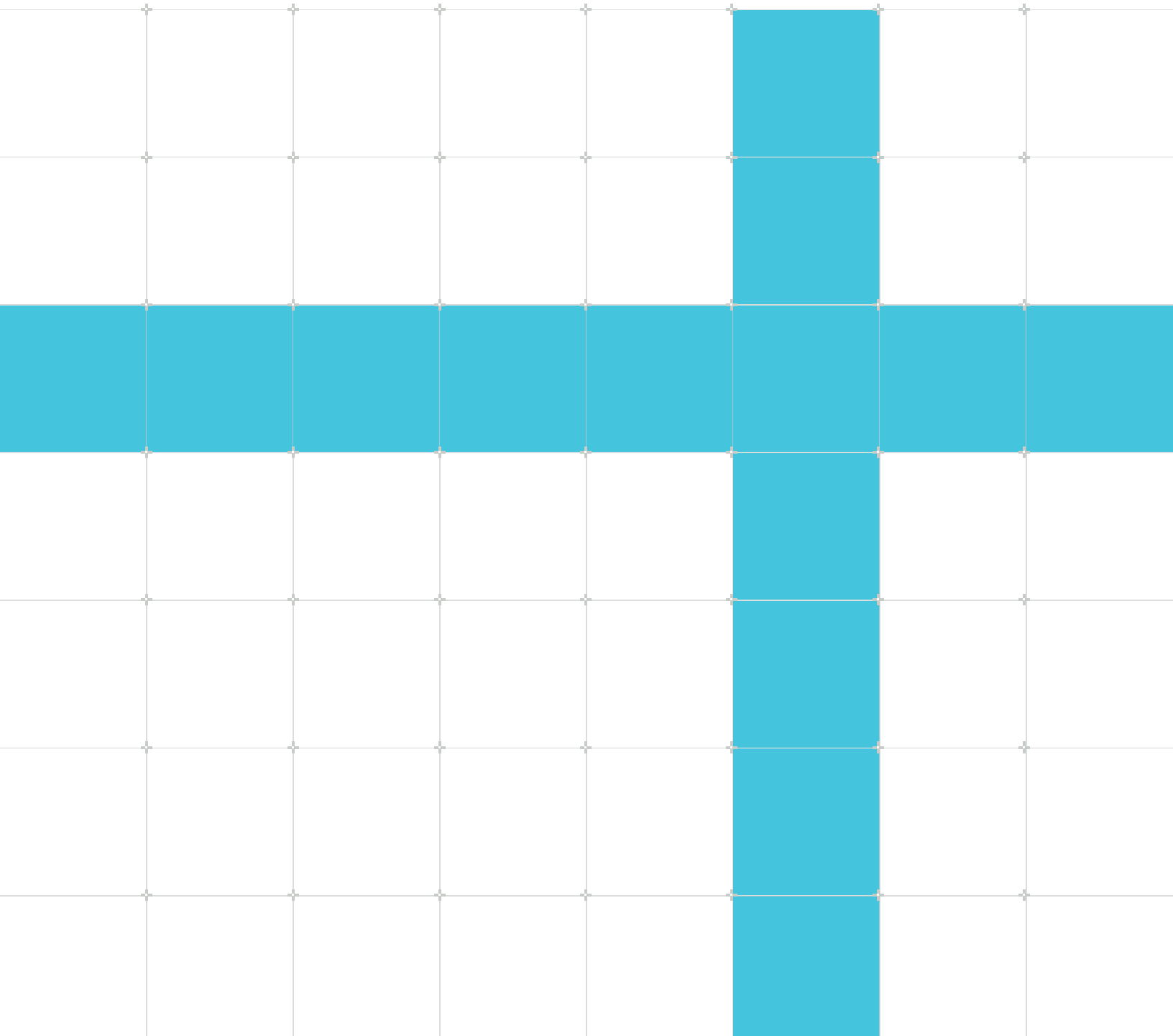
Arm® CoreLink™ GIC-720AE Generic Interrupt Controller

Revision: r2p0

Technical Reference Manual

Non-Confidential

Issue 10
102666_0200_10_en



Arm® CoreLink™ GIC-720AE Generic Interrupt Controller

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0200-10	28 February 2025	Non-Confidential	Third early access release for r2p0
0100-09	31 December 2024	Non-Confidential	Third early access release for r1p0
0200-08	19 July 2024	Non-Confidential	Second early access release for r2p0
0100-07	23 May 2024	Non-Confidential	Second early access release for r1p0
0001-06	12 April 2024	Confidential	Third early access release for r0p1
0200-05	29 February 2024	Confidential	First early access release for r2p0
0001-04	19 December 2023	Confidential	Second early access release for r0p1
0100-03	3 November 2023	Confidential	First early access release for r1p0
0001-02	18 September 2023	Confidential	First early access release for r0p1
0000-01	4 August 2023	Confidential	First early access release for r0p0

Proprietary Notice

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

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

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

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

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

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

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

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Feedback

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

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

Inclusive language commitment

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

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

Contents

1. Introduction

1.1 Product revision status

The r*x*p*y* identifier indicates the revision status of the product described in this manual, for example, r1p2, where:

r*x* Identifies the major revision of the product, for example, r1.

p*y* Identifies the minor revision or modification status of the product, for example, p2.

1.2 Intended audience

This book is written for system designers and programmers who are designing or programming a *System on Chip* (SoC) that uses the GIC-720AE.

1.3 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Convention	Use
italic	Citations.
bold	Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`

Convention	Use
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

We recommend the following. If you do not follow these recommendations your system might not work.

Your system requires the following. If you do not follow these requirements your system will not work.

You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.

This information is important and needs your attention.

This information might help you perform a task in an easier, better, or faster way.

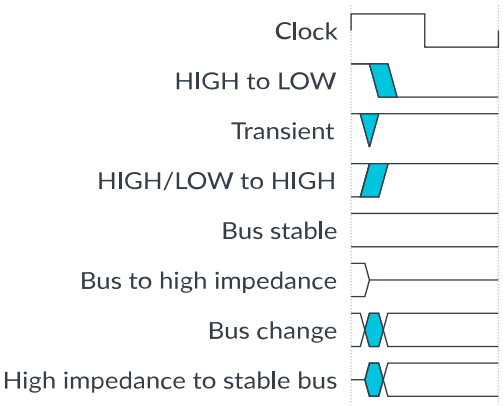
This information reminds you of something important relating to the current content.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

Figure 1-1: Key to timing diagram conventions



Signals

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name, n denotes an active-LOW signal.

1.4 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® CoreLink™ ADB-400 AMBA® Domain Bridge User Guide	DUI 0615F	Confidential
Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual	102667_0200_10_en	Confidential
Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Development Interface Report	102669_0200_06_en	Confidential
Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Safety Manual	102668_0200_06_en	Confidential
Arm® GIC MSI Delivery Interface	AES 0019A	Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® APB Protocol Specification	IHI 0024D	Non-Confidential
AMBA® AXI Protocol Specification	IHI 0022J	Non-Confidential
AMBA® AXI-Stream Protocol Specification	IHI 0051B	Non-Confidential
AMBA® Low Power Interface Specification	IHI 0068D	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487K.a	Non-Confidential
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069H.b	Non-Confidential
Arm® Generic Interrupt Controller v3 and v4 - Virtualization	107627	Non-Confidential
Learn the architecture - Arm® Generic Interrupt Controller v3 and v4	198123	Non-Confidential
Locality-Specific Peripheral Interrupts, Arm® Generic Interrupt Controller v3 and v4	102923	Non-Confidential

Non-Arm resources	Document ID	Organization
Standard Manufacturer's Identification Code	JEP106BJ	JEDEC, https://www.jedec.org/

Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com.

2. About the GIC-720AE

The GIC-720AE is a *Functional Safety* (FuSa) variant of the GIC-700. The GIC-720AE is a *Generic Interrupt Controller* (GIC) that handles interrupts from peripherals to the cores and between cores. The GIC-720AE supports a distributed microarchitecture containing several individual blocks that are used to provide a flexible GIC implementation.

The GIC-720AE supports the GICv3, GICv3.1, and GICv4.1 architecture. For more information, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

The microarchitecture scales from a single core to coherent multichip environments containing up to 16 chips of up to 512 cores each.


Note

This manual defines a *chip* as an SoC that is integrated with the GIC-720AE. A single-chip system has one SoC. A multichip system can have several SoCs that are connected externally, or an SoC comprising several SoCs connected inside a single physical package. In all cases, each SoC is integrated with the GIC-720AE.

All the GIC-720AE blocks communicate through fully credited AXI5-Stream interface channels. Therefore, an interface only exerts transient backpressure on its ic<xy>tready signals, enabling packets to be routed over any free-flowing interconnect. Channels can be routed over dedicated AXI5-Stream buses, or over any available free-flowing transport layer in the system. A channel is described as free-flowing when all transactions on that channel complete without a non-transient dependency on any other transaction.

The GIC-720AE includes build scripts that can create appropriate levels of hierarchy for any particular configuration.


Note

The GIC-700 information is unchanged, and information about the FuSa features available in GIC-720AE can be found in 6. Functional safety in GIC-720AE on page 331.

2.1 Component overview

The GIC-720AE comprises several significant blocks that work in combination to create a single architecturally compliant GICv3, GICv3.1, and GICv4.1 implementation within the system.

Each significant block contains lock-stepped primary and secondaries to protect the blocks core logic, plus some protection mechanisms that protect the interfaces of the block.

The GIC-720AE consists of the following blocks:

Distributor (GICD)

The Distributor is the hub of all the GIC communications and contains the functionality for all *Shared Peripheral Interrupts* (SPIs), real-time SPIs, and also *Locality-specific Peripheral Interrupts* (LPIs). The Distributor supports up to 960 real-time SPIs, to use with devices that require a deterministic interrupt latency response. It is responsible for the entire GIC programmers model, except for the GITS_TRANSLATER register, which is hosted in the *Interrupt Translation Service* (ITS) block.
In configurations that support GICv4.1, the Distributor also manages vSGIs and the management of vPEs.

The Distributor also maintains the coherency of the SPI register space in multichip configurations.

The LPI functionality for all cores on a chip is combined into a single cache in the Distributor.

GIC Cluster Interface (GCI)

The GCI maintains the *Private Peripheral Interrupts* (PPIs) and *Software Generated Interrupts* (SGIs) for a particular set of cores. A GCI can scale from 1-64 cores and is best placed next to the processors that it is servicing to reduce wiring to the cores.
A GCI is also referred to as a Redistributor.

The GICv3 and v4.1 architecture specifies a Redistributor address space containing two pages for each core for GICv3 and four pages for each core for GICv4.1. The SGI page functionality is contained in the GIC-720AE Redistributor. However, the Distributor contains the other pages for all cores on a chip. To ensure that the GCI receives compliant GIC Stream protocol from the cores, it contains a *GIC Stream Protocol Validator* (GSPV).

The GIC-720AE supports powering down the GCIs and the associated cores, separately from the Distributor.

During configuration, the GCI can be set to provide a wake request signal for each of the cores it supports.

Interrupt Translation Service (ITS)

The ITS translates message-based interrupts, *Message-Signaled Interrupts* (MSI/MSIx), from an external *PCI Express* (PCIe) *Root Complex* (RC), or other sources. The ITS also manages LPIs during core power management.
The GIC-720AE supports up to 32 ITS blocks for each chip.

For more information about the ITS, see the *Locality-Specific Peripheral Interrupts, Arm® Generic Interrupt Controller v3 and v4*.

MSI-64 Encapsulator

The MSI-64 Encapsulator is a small block that combines the *DeviceID* (DID), required by writes to the GITS_TRANSLATER register, into a single memory access.

SPI Collator

The GIC-720AE supports up to 1984 SPIs that are spread across the system, but this quantity reduces by the number of real-time SPIs that connect to the Distributor. The SPI Collator enables standard SPIs to be converted into messages remotely from the Distributor.

Up to 32 SPI Collators can be supported in a single configuration. The 1984 SPIs minus any real-time SPIs, can be spread across 32 SPI Collators, with a maximum of 1024 standard SPIs in one SPI Collator.

Wake Request

The Wake Request contains all the architecturally defined wake_request signals for each core on the chip. It is a separate block that can be positioned remotely from the Distributor, such as next to a system control processor.

GIC interconnect

The GIC interconnect is a set of components that can be used for routing the AXI5-Stream interfaces between the different blocks.

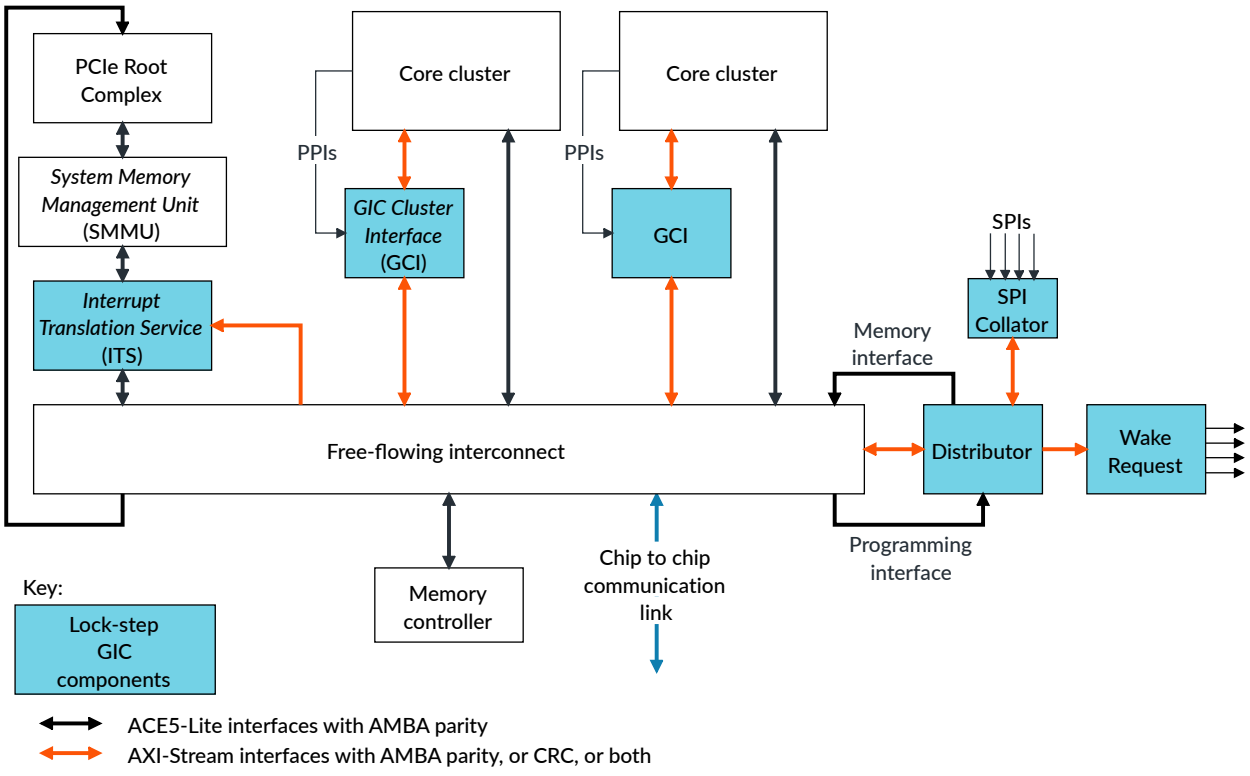
Top level

The top level has no specific interfaces but combines the interfaces of other blocks within the clock or power domain to reduce the number of domain bridges. The GIC-720AE build scripts enable you to build the GIC from either:

- A single combined block that uses a dedicated 16-bit AXI5-Stream interconnect.
- A set of individual blocks that interconnect using your own transport layer.

The following figure shows a GIC-720AE with a free-flowing interconnect in an example system.

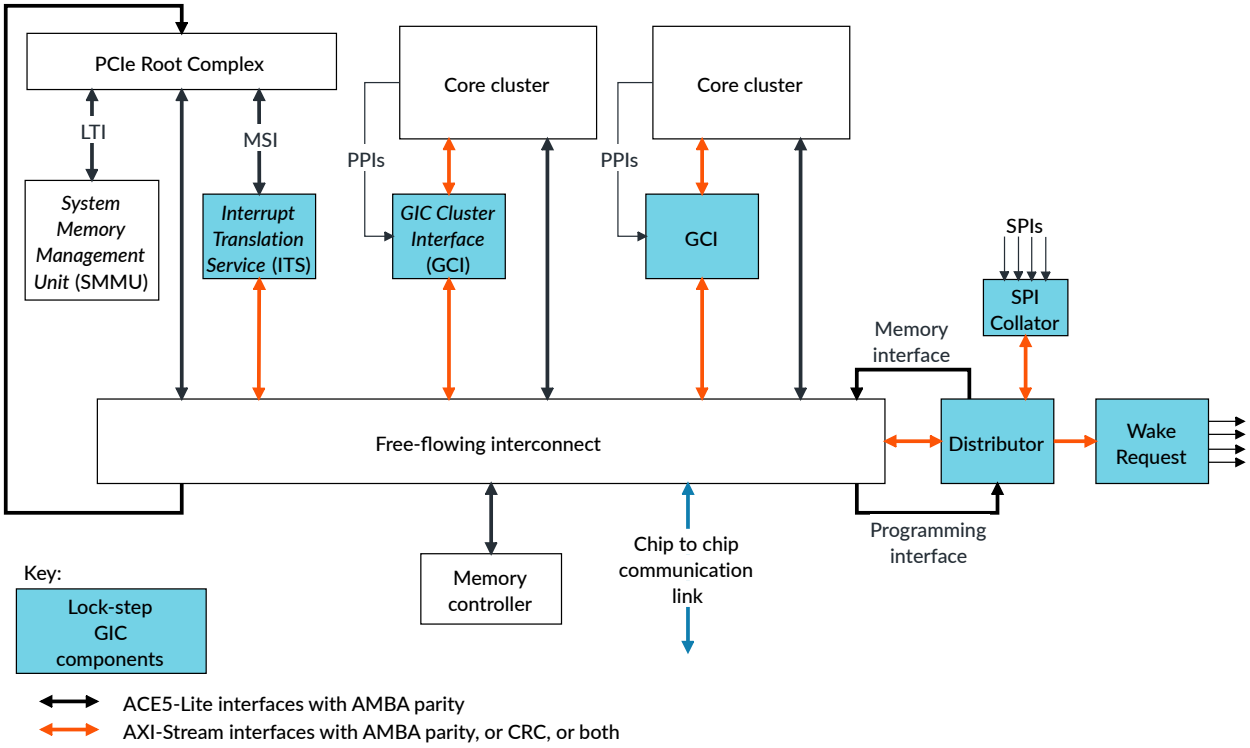
Figure 2-1: GIC-720AE with free-flowing interconnect in an example system



A free-flowing channel is clear to transmit a transaction that arrives at its destination without any non-transient dependencies on other transactions.

The following figure shows a GIC-720AE example system with the PCIe root complex connecting directly to the interconnect.

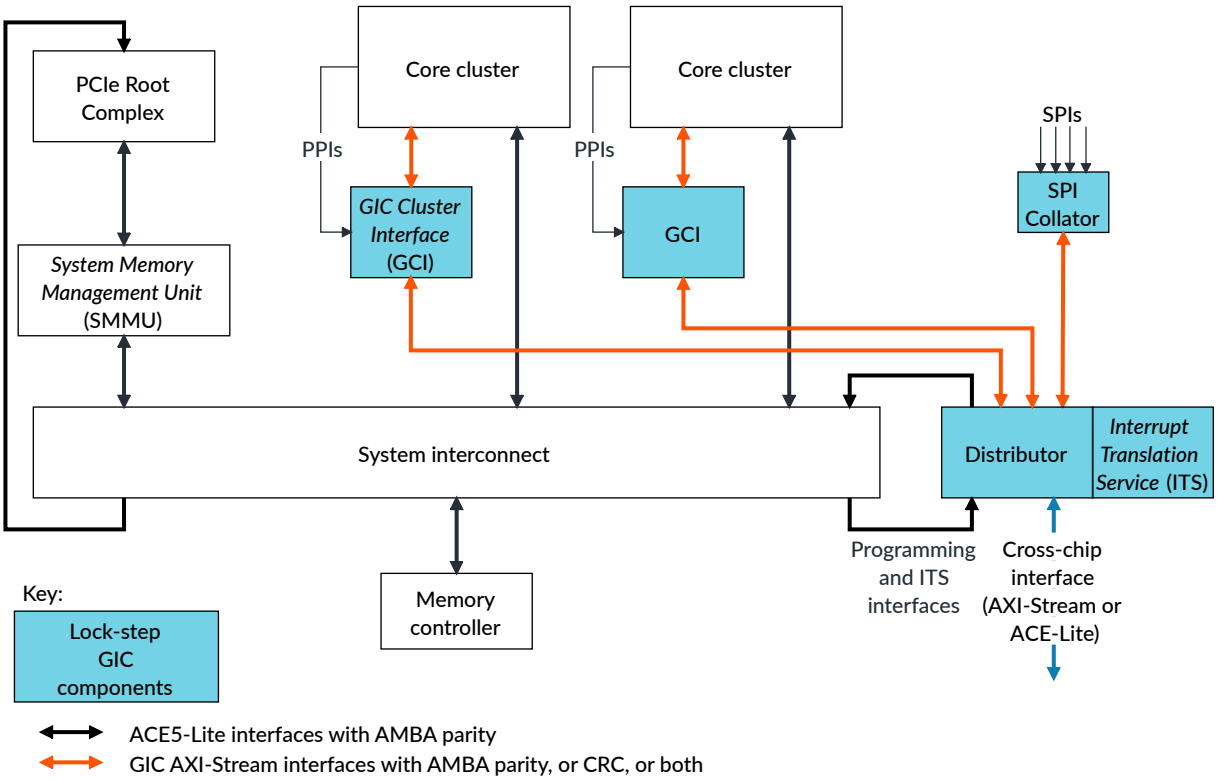
Figure 2-2: GIC-720AE with interconnect in an example system



Cross-chip interfaces enable communication between cores in a multichip configuration.

The following figure shows a monolithic GIC-720AE with interconnect in an example system.

Figure 2-3: Monolithic GIC-720AE with interconnect in an example system



If the GIC supports LPIs, there must be free-flowing access to main memory. This requirement is irrespective of the interconnect that is used for routing the AXI5-Stream interfaces. For more information, see the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual* and the interconnect documentation.

The GIC-720AE implements version 3, 3.1, and 4.1 of the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*. To use GIC-720AE with a core, the core must:

- Implement any of the Armv8.x-A or Armv8.x-R architectures and support the GIC Stream protocol.
- Support the extended range of GICv3.1 interrupts, when GIC-720AE is configured and programmed to use >960 SPIs or >16 PPIs for each core.
- Support GICv4.1, when GIC-720AE is configured and programmed to use the GICv4.1 features.
- Support AMBA parity on its AXI5-Stream interfaces.

2.2 Compliance

The GIC-720AE interfaces are compliant with Arm specifications and protocols.

The GIC-720AE is compliant with:

- The AMBA® AXI5-Stream protocol. See the *AMBA® AXI-Stream Protocol Specification*.
- The AMBA ACE5-Lite protocol. See the *AMBA® AXI Protocol Specification*.
- Version 3.1 and 4.1 of the Arm GIC architecture specification. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.
- The *Arm® GIC MSI Delivery Interface*.
- The GIC Stream protocol. See the *GIC Stream Protocol interface* appendix in the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

2.3 Features

The GIC-720AE provides interrupt services and masking, registers and programming, interrupt grouping, security, performance monitoring, and error correction and detection.

Interrupt services and masking

The GIC-720AE provides the following interrupt features:

- Support for the following interrupt types:
  - Up to 56000 physical *Locality-specific Peripheral Interrupts* (LPIs). A peripheral generates these interrupts by writing to a memory-mapped register in the GIC-720AE.
  - Direct injection of up to 56000 virtual LPIs for each *virtual processing element* (vPE), when the GIC is configured to support GICv4.1.
  - Up to 1984 *Shared Peripheral Interrupts* (SPIs) in groups of 32. Up to 960 of these 1984 SPIs can be assigned as real-time SPIs, to use with devices that require a deterministic interrupt latency response.
  - Up to 48 *Private Peripheral Interrupts*) (PPIs) that are independent for each core and can be programmed to support either edge-triggered or level-sensitive interrupts. If a *GIC Cluster Interface* (GCI) is configured to support real-time interrupts, then these PPIs have a deterministic interrupt latency response for the PEs on that GCI.
  - Up to 16 physical *Software Generated Interrupts* (SGIs) for each core, which the core generates through its GIC CPU interface. If a GCI is configured to support real-time interrupts, then these SGIs have a deterministic interrupt latency response for the PEs on that GCI.
  - Direct injection of up to 16 virtual SGIs for each vPE, when the GIC is configured to support GICv4.1.
  - Provides four logical views so that up to three different OSs within a system can be assigned a different view.
- Up to 32 *Interrupt Translation Service* (ITS) modules that provide device isolation and ID translation for message-based interrupts and enable virtual machines to program devices directly.
- Interrupt masking and prioritization with 32 priority levels, 5 bits for each interrupt.

Registers and programming

The GIC-720AE provides the following programming features:

- Flexible affinity routing, using the *Multiprocessor Identification Register* (MPIDR) addresses, including support for four affinity levels (0-3).
- Single ACE5-Lite subordinate interface on each chip for programming of all registers but excluding the GITS_TRANSLATER register in non-monolithic configurations. Each ITS has an optional ACE5-Lite subordinate interface for programming the GITS_TRANSLATER register.
- Coherent view of SPI register data across multiple chips.

Security

The GIC-720AE provides the following security features:

- A global *Disable Security* signal. The gicd_ctlr_ds signal enables support for systems without security support. If the GIC configuration supports multi view, then the GIC assigns one gicd_ctlr_ds[3:0] signal bit to each view.
- The following interrupt groups allow interrupts to target different Exception levels:
  - Group 0
  - Non-secure Group 1
  - Secure Group 1

  See 4.3 Interrupt groups and security on page 59 for more information about security and groupings.

For more information about Exception levels, see the *Arm® Architecture Reference Manual for A-profile architecture*.

Performance monitoring

The GIC-720AE provides *Performance Monitoring Unit* (PMU) counters with snapshot functionality.

Error correction and containment

The GIC-720AE provides the following error correction features:

- Armv8.2 *Reliability Accessibility Serviceability* (RAS) architecture-compliant error reporting for:
  - Software access errors.
  - ITS command and translation errors.
  - *Error Correcting Code* (ECC) errors.
- Containment of errored interrupts, to enable software recovery where possible.
- Software mechanism to trigger and test the error recovery functionality.
- *GIC Stream Protocol Validator* (GSPV), which ensures that the *GIC Cluster Interface* (GCI) receives compliant AXI5-Stream protocol and GIC Stream protocol.

The PMU and RAS error records are in the GICP and GICT register spaces, respectively. If the Security state changes, these registers retain their contents unless the debug reset signal (dbg_reset_n) goes LOW.

Error detection

GIC-720AE contains the following error detection features:

- Lock-step of GIC blocks.
- AMBA parity on ACE5-Lite, AXI5-Stream, APB5, Q-Channel and P-Channel interfaces.
- Duplicated reset and clock with consistency detection.
- Protected MBIST interface signals.
- DFT false activation detection.
- Duplicated interrupt and interrupt return wires with consistency detection.
- Interrupt wire, and check wire, consistency protection using lock-step, flop parity, and BIST.
- Real-time interrupt prioritization protection using a combination of techniques including lock-step, flop parity, duplication, and output checking using alternate logic.
- End-to-end CRC protection over AXI5-Stream.
- End-to-end CRC protection over the cross-chip interface, which is configured to be either AXI5-Stream or ACE5-Lite.

2.4 Test features

The GIC-720AE provides *Design for Test* (DFT) signals for test mode.

2.5 Product documentation

Documentation that is provided with this product includes a *Technical Reference Manual* (TRM), a *Configuration and Integration Manual* (CIM), a *Safety Manual*, and a *Development Interface Report*.

For relevant protocol and architectural information that relates to this product, see 1.4 Useful resources on page 15.

The GIC-720AE documentation is as follows:

Technical Reference Manual

The TRM describes the functionality and the effects of functional options on the behavior of the GIC-720AE. It is required at all stages of the design flow. The choices that are made in the design flow can mean that some behaviors that the TRM describes are not relevant. If you are programming the GIC-720AE, contact:

- The implementer to determine:
  - The build configuration of the implementation
  - What integration, if any, was performed before implementing the GIC-720AE
- The integrator to determine the signal configuration of the device that you use

The TRM complements architecture and protocol specifications and relevant external standards. It does not duplicate information from these sources.

Configuration and Integration Manual

The CIM describes:

- The available build configuration options
- How to configure the *Register Transfer Level* (RTL) with the build configuration options
- How to integrate the GIC-720AE into an SoC
- How to implement the GIC-720AE into your design
- The processes to validate the configured design

The Arm product deliverables include reference scripts and information about using them to implement your design.

The CIM is a confidential document that is only available to licensees.

Safety Manual

The SM provides additional information on specific features of the GIC-720AE that are relevant to Functional Safety. This information is important for SoC integrators whose final designs target applications where Functional Safety is a concern.
The SM is a confidential document that is only available to licensees.

Development Interface Report

The DIR describes the activities conducted by Arm that are related to the safety architecture of the GIC-720AE.
The DIR is a confidential document that is only available to licensees.

2.6 Product revisions

This section describes the differences in functionality between product revisions.

r0p0 First release.

r0p0-r0p1 The functional changes are:

- Bug fixes.

r0p1-r1p0 The functional changes are:

- Added the multi view feature. See 4.5 Multi view on page 63 for more information.
- Added *GIC Stream Protocol Validator* (GSPV), which enables a GCI to connect to a processor with a safety level that is lower than ASIL-D. See 3.2 GIC Cluster Interface on page 37.
- Added CRC protection to the cross-chip ACE5-Lite interface.

r1p0-r2p0 The functional changes are:

- Added real-time interrupts. See 4.8 Low latency support on page 70 for more information.

3. Components in GIC-720AE

The GIC-720AE contains several major components that use an internal GIC interconnect to route the AXI5-Stream interfaces between the different components. A configuration parameter controls the hierarchy of the GIC components.

The components are:

- Distributor
- *GIC Cluster Interface* (GCI)
- *Interrupt Translation Service* (ITS)
- MSI-64 Encapsulator
- SPI Collator
- Wake Request
- GIC interconnect

Each component is configurable so that it can be modified for the system requirements.

The hierarchy of the GIC components can be a single combined block that uses a dedicated 16-bit or 64-bit AXI5-Stream interconnect, or a set of individual blocks that are interconnected using your own transport layer.

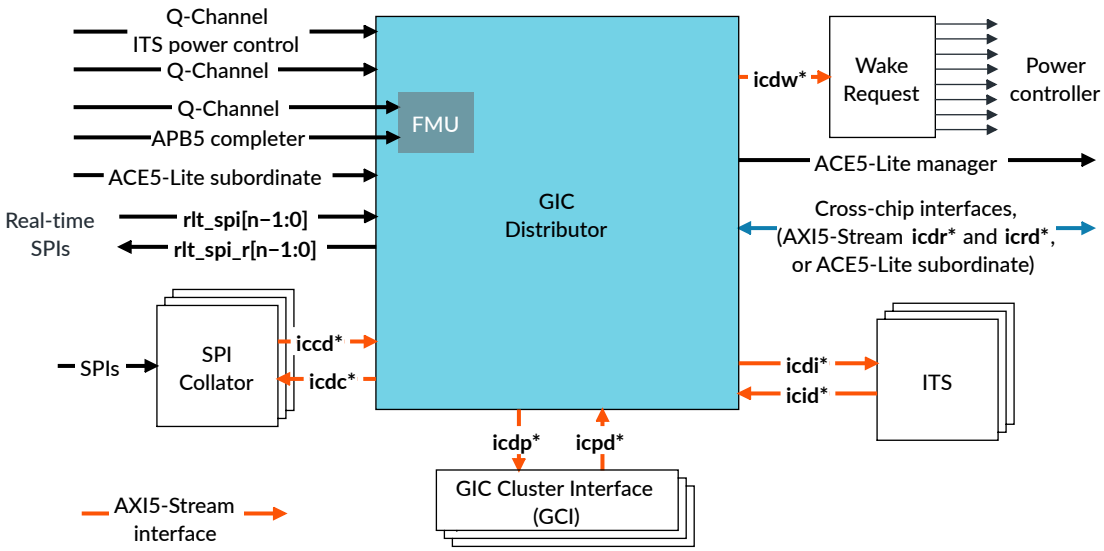
3.1 Distributor (GICD)

The Distributor is the main communication point between all GIC-720AE blocks. It performs SPI management, real-time SPI prioritization, and LPI caching, and all communications with other blocks and chips. It also contains the *Fault Management Unit* (FMU).

When the GIC is configured to support GICv4.1, then the Distributor also performs vPE management.

The following figure shows the Distributor and its interfaces. The figure does not show the protection signals that a configuration can include.

Figure 3-1: GIC-720AE Distributor



Note

If GICD_CFGID.ACE_CC == 1, the GIC replaces the icdr* and icrd* AXI5-Stream interfaces with an ACE5-Lite subordinate interface.

The Distributor is the main hub of the GIC and it implements most of the GICv4.1 architecture including:

- Programming, forwarding, and prioritization of SPIs and real-time SPIs, see 4.11 SPIs on page 75.
- Caching and forwarding of LPIs, see 4.13 LPIs on page 86.
- SGI routing and forwarding, see 4.9 SGIs on page 71.
- vSGI forwarding and routing, when the GIC is configured to support GICv4.1.
- Management and control of vPEs and residency, when the GIC is configured to support GICv4.1.
- Programming interface for all registers, apart from GITS_TRANSLATER.
- Power control of cores and Redistributors.

The Distributor also contains the FMU, which processes the faults that the protection mechanisms detect from all GIC blocks. See 6.1 Fault Management Unit on page 333 for more information.

I/O hub support

The Distributor supports configurations with no *GIC Cluster Interfaces* (GCIs). This option is for systems where a central hub device has interrupt I/O and a Distributor that delivers interrupts to remote compute chiplets. Distributors with LPI support but no ITSs are also supported for these systems, where LPIs are only generated on the central hub device. Software can read

GICD_CFGID.NITS to determine if no local ITSs are present. If a Distributor has no GCI, then it includes the GICD_VCFGBASER and GICD_VSLEEPR registers.

If a system contains a chip with no GCIs, then 1 of N SPIs cannot be supported. Therefore, all GIC configurations in the system must disable 1 of N support by setting `spi_1ofn_support` = 0 in each configuration.

3.1.1 Distributor AXI5-Stream interfaces

The GIC-720AE uses AXI5-Stream interfaces to communicate between blocks.

These interfaces are:

- Fully credited
- ic<xy>tready. Where xy can be cd, dc, pd, dp, id, di, rd, dr, or dw.

Irrespective of the interconnect that is used, packets must not be reordered between endpoints, for example, between the Distributor and a single Redistributor block. Packets must never be interleaved.

The number of credits, or the outstanding transaction capability, is fixed across all the AXI5-Stream interfaces with the following exceptions:

- The number of outstanding LPIs from each ITS to the GICD can be set using the `number_int_credit` (1-16) and `number_ll_int_credit` (0-4) configuration parameters, for transactions that have been locked in the ITS caches using the GITS_OPR register.
- The total number of LPIs and vLPIs transfers that can be outstanding from one chip to another chip, can be set from 1-8 with the `lpi_cc_tokens` configuration parameter.
- The total number of SGIs that can be in transit from one chip to another chip, can be set from 1-8 with the `sgi_cc_tokens` configuration parameter.
- The total number of vSGIs that can be in transit from one chip to another chip, can be set from 1-8 with the `vsgi_cc_tokens` configuration parameter.
- The GICD_FCTLR3 can set an overall limit on the number of transactions for the cross-chip AXI5-Stream interfaces. If the cross-chip interface is configured to use ACE5-Lite, then software can use GICD_CCCTLR to limit the number of transactions.

For information about AXI5-Stream signals, see the *AMBA® AXI-Stream Protocol Specification*.

The following table lists the AXI5-Stream input interfaces.

Table 3-1: AXI5-Stream input interface descriptions

Bus	Destination	Width	ic<xy>tid
ICID	ITS to Distributor	16-bit or 64-bit	ITS number
ICPD	Redistributor to Distributor	16-bit, 32-bit, or 64-bit	Redistributor number
ICCD	SPI Collator to Distributor	16-bit	SPI Collator number
ICRD	Remote chip to Distributor	64-bit	0

The following table lists the AXI5-Stream output interfaces.

Table 3-2: AXI5-Stream output interface descriptions

Bus	Destination	Width	ic<xy>tdest
ICDI	Distributor to ITS	16-bit or 64-bit	ITS number
ICDP	Distributor to Redistributor	16-bit, 32-bit, or 64-bit	Redistributor number
ICDC	Distributor to SPI Collator	16-bit	SPI Collator number
ICDR	Distributor to remote chip	64-bit	Programmed value
ICDW	Distributor to Wake Request block	16-bit	-

Each bus has an associated ic<xy>twakeup signal that requests wakeup through the qactive signals when the Distributor, or destination block, is hierarchically clock gated through the Q-Channel. The ic<xy>twakeup input signal must be driven from a cleanly registered version of ic<xy>tvalid, to prevent spurious wake ups from any signal glitches.

For information about the Distributor Q-Channels, see 3.1.4 Distributor Q-Channels on page 34.

3.1.2 Distributor ACE5-Lite subordinate interface

The AMBA® ACE5-Lite subordinate port on the GIC-720AE Distributor provides access to the entire register map except for the GITS_TRANSLATER register. The interface supports 64-bit, 128-bit, 256-bit, or 512-bit data widths.

The GIC-720AE only accepts single beat accesses of the sizes for each register that are shown in the programmers model, see 5. Programmers model for GIC-720AE on page 132.

When the GIC-720AE is a monolithic configuration without MSI-64 support, the Distributor and ITS both share an ACE5-Lite subordinate port, and the DeviceID for the ITS translation is taken from the awuser_s[did_width−1:0] signal. The value of the `did_width` parameter is set during silicon integration. For more information about the ITS, see 3.3 Interrupt Translation Service on page 41.

The following table shows the acceptance capabilities of the Distributor ACE5-Lite subordinate interface. These acceptance capabilities also apply to the cross-chip ACE5-Lite subordinate interface.

Table 3-3: Distributor ACE5-Lite subordinate interface acceptance capabilities

Attribute	Capability
Combined acceptance capability	3
Read acceptance capability	2
Read data reorder depth	1
Write acceptance capability	2

The GIC-720AE uses awatop_s, a<x>cache_s, a<x>domain_s, and a<x>snoop_s signals to detect cache maintenance operations that are responded to in a protocol-compliant manner but are

otherwise ignored. The GIC-720AE also ignores other Cacheability, Shareability, and protection settings, except for the a<x>prot_s[1] security signal.

If you are connecting to an AXI3 or AXI4 port, then awatop_s, a<x>domain>_s, a<x>snoop_s, and, for AXI3, a<x>len[7:4] signals must all be tied LOW.

The GIC-720AE uses the wstrb signal to determine the size of a transaction. The GIC rejects transactions where the strobes do not form a continuous block that is address aligned with the resultant size of the transaction.

The GIC-720AE has a separate awakeup_s signal to force the GIC to wakeup when it is hierarchically clock gated through the Q-Channel. The awakeup_s signal must be connected to a cleanly registered version of (awvalid_s | arvalid_s signal) to ensure that the GIC does not request to be woken up due to incoming signal glitches.

The GIC-720AE address map has multiple pages. The number of pages and the address aliasing depends on your configuration. See 5.1 Register map pages on page 132.

You must set up the system address map so that each core accesses the GICD page on its local chip at the same address. All other pages must be globally accessible, although access of pages on a remote chip by a core is expected to be rare.

3.1.2.1 SLVERR error cases

The GIC ignores any transactions that are not standard single-beat memory accesses to a defined register, and it responds in a protocol-compliant manner.

If the GIC receives an errant transaction, then it records the error in software error record (Record 0). If GICT_ERR0CTLR.UE =1, the GIC returns an SLVERR response to an errant transaction. These error responses are disabled by default from reset. Software can disable some error reporting such as out-of-range register or accesses to unimplemented SPI registers, by using the GICT_ERR0CTLR.DIS_* bits.

Note

The subordinate interface does not support dataless cache stash transactions so they must not target the GIC.

It is also possible when accessing SPI, PPI, or SGI registers that data corruption might occur in the memory. If the internal ECC protection detects corrupt data, then it records the error in error record 0. The values in GICT_ERR0CTLR.UE and GICD_FCTLR2.ARP control how the GIC reports the error to the system, as the following table shows.

Table 3-4: Subordinate response signaling for ECC detection errors

GICT_ERR0CTLR.UE	GICD_FCTLR2.ARP	ACE signal
0	0	None
1	0	rresp signal returns SLVERR

GICT_ERR0CTLR.UE	GICD_FCTLR2.ARP	ACE signal
X	1	rpoison signal is HIGH

GICD_FCTLR2.AWP controls whether the GIC uses the wpoison signal (causing the GIC to reject the transaction and report it) or whether the GIC ignores wpoison.

The GIC never returns a DECERR response.

3.1.2.2 AMBA bus properties, GICD and CC subordinate interfaces

The AMBA® protocols define multiple property types that indicate the capabilities of a device.

The cross-chip (CC) subordinate interface only accepts INCR or aligned WRAP transactions. Also, 64-bit atomicity is required between the CC manager interface and the destination CC subordinate interface. Therefore, for any split transactions, the address must update to correctly align the data.

The following table lists the ACE5-Lite properties for the GICD subordinate interface and the cross-chip subordinate interface.

Table 3-5: GICD and cross-chip ACE5-Lite subordinate interface properties

AMBA property	Subordinate interface	ACE5-Lite issue
Atomic_Transactions	Ignore and respond legally	F
Barrier_Transactions	False	F
Cache_Stash_Transactions	Basic when `axi_cache_stashing_support` == 0. Full cache stash support, including dataless, when `axi_cache_stashing_support` == 1. Ignore and respond legally.	F
Check_Type	False, Odd_Parity_Byte_All	F
CMO_On_Read	Ignore and respond legally	G
CMO_On_Write	False	G
Coherency_Connection_Signals	False	F
DeAllocation_Transactions	Ignore and respond legally	F
DVM_v8	False	F
DVM_v8.1	False	F
DVM_v8.4	False	H
DVM_v9.2	False	J
Exclusive_Accesses	False	F
InvalidateHint_Transaction	Ignore and respond legally	J
Loopback_Signals	True	F
Max_Transaction_Bytes	4096	F
MPAM_Support	False	G
MTE_Support	Ignore and respond legally	H
NSAccess_Identifiers	False	F

AMBA property	Subordinate interface	ACE5-Lite issue
Persist_CMO	Ignore and respond legally	F
Poison	True	F
Prefetch_Transaction	False	H
QoS_Accept	False	F
Read_Data_Chunking	True	G
Read_Interleaving_Disabled	No read data interleaving	G
RME_Support	True when `axi_rme_support` == 1	J
Shareable_Transactions	True	F
Trace_Signals	True	F
Unique_ID_Support	True	G
Untranslated_Transactions	False	F
Wakeup_Signals	True	F
Write_Plus_CMO	False	H
WriteEvict_Transaction	True	F

3.1.3 Distributor ACE5-Lite manager interface

The GICD uses the AMBA® ACE5-Lite manager interface to access all pending, property, and translation tables that are allocated to the GIC. This interface is only present when LPIs are supported, or the GIC has an ACE5-Lite cross-chip interface, or both.

The interface can be configured to be 64-bit, 128-bit, 256-bit, or 512-bit wide.

For multichip configurations, if the GIC has an ACE5-Lite cross-chip interface, then it uses the GICD ACE5-Lite manager interface for cross-chip communications. The system must ensure that traffic from the GICD ACE5-Lite manager interface can reach the cross-chip ACE5-Lite subordinate interfaces of other GICDs in the system, in a free-flowing way without blocking access to memory.

If the cross-chip interface is configured to use ACE5-Lite, then the GICD_CCCTLR register provides options to control the ACE5-Lite traffic between chips.

The following table shows the issuing capabilities of the Distributor ACE5-Lite manager interface.

Table 3-6: Distributor ACE5-Lite manager interface issuing capabilities

Attribute	Capability	
	Read	Write
8-bit reads to Property table (physical or virtual)	9	0
8-bit read or write to the Pending table (physical or virtual)	2	2
Accesses to ITS tables, 64-bit or less	sum(mpfa_counts of all ITSs)	Number of ITS
256-bit read of ITS command queue	Number of ITS	0

Attribute	Capability	
	Read	Write
512-bit accesses of Pending tables (physical or virtual)	1	1
256-bit accesses of Pending tables or Property tables	2	2
Accesses to vPE Configuration table or vPT, 256-bit or less	3	3
Cross-chip transactions	-	Set by the `ace_cc_credits` parameter

Each transaction uses a unique transaction ID.

The following GIC registers are shared between Redistributors, and these registers must be set to the same value by each core that has enabled LPIs:

- GICR_PROPBASER
- GICR_PENDBASER, but excluding the ADDRESS field
- GICR_VPROPBASER and GITS_BASERn, in configurations that support GICv4.1

The ACE5-Lite manager interface cannot issue barriers or *Cache Maintenance Operations* (CMOs). However, it can issue shareable, ReadOnce and WriteUnique, transactions if programmed to do so.

3.1.3.1 AMBA bus properties, GICD manager interface

The AMBA® protocols define multiple property types that indicate the capabilities of a device.

The following table lists the Distributor ACE5-Lite manager interface properties.

Table 3-7: GICD ACE5-Lite manager interface properties

AMBA property	Manager interface	ACE5-Lite issue
Atomic_Transactions	False	F
Barrier_Transactions	False	F
Cache_Stash_Transactions	False	F
Check_Type	False, Odd_Parity_Byte_All	F
CMO_On_Read	False	G
CMO_On_Write	False	G
Coherency_Connection_Signals	False	F
DeAllocation_Transactions	False	F
DVM_v8	False	F
DVM_v8.1	False	F
DVM_v8.4	False	H
DVM_v9.2	False	J
Exclusive_Accesses	Not used	F
InvalidateHint_Transaction	False	J

AMBA property	Manager interface	ACE5-Lite issue
Loopback_Signals	False	F
Max_Transaction_Bytes	64	F
MPAM_Support	Support as defined by the GIC architecture.	G
MTE_Support	False	H
NSAccess_Identifiers	False	F
Persist_CMO	False	F
Poison	True	F
Prefetch_Transaction	False	H
QoS_Accept	False	F
Read_Data_Chunking	True	G
Read_Interleaving_Disabled	Read data interleaving is accepted.	G
Regular_Transactions_Only	True for GICD manager interface. When `ace_cc` == 1, the cross-chip traffic uses this interface, so Regular_Transactions_Only is false. When `ace_cc` == 0 and only memory traffic uses this interface, Regular_Transactions_Only is true.	H
RME_Support	True when `axi_rme_support` == 1.	J
Shareable_Transactions	Not used	F
Trace_Signals	False	F
Unique_ID_Support	True	G
Untranslated_Transactions	False	F
Wakeup_Signals	True	F
Write_Plus_CMO	False	H
WriteEvict_Transaction	False	F

The manager interface does not issue fixed bursts.

3.1.4 Distributor Q-Channels

There is a single Q-Channel for clock gating the GIC-720AE Distributor. The Q-Channel interface denies access when the Distributor is busy processing interrupts.

The Distributor also has a separate Q-Channel that enables power control for each configured ITS. The GIC only accepts a low-power request when GITS_CTLR.Quiescent is set. If the Quiescent bit is set, the Q-Channel qacceptn_its_<n> signal is asserted, and the GIC guarantees that the bus to the relevant ITS is idle in both directions and that the ITS can be powered down. To perform wake-on-LPI functionality, you can use GITS_FCTLR.PWE to disable the bus while the ITS is still active and able to translate interrupts. If the bus is disabled, then when the qactive_gicd signal asserts on the corresponding ITS, the system must re-enable the bus and program the GICD so that it is ready to receive LPIs. The system must route the qactive_gicd signal to a power controller that implements the following sequence:

1. Power up the GICD.
2. Restore the GICD program state.
3. Turn on the associated ITS Q-Channel on the GICD, which allows the ITS to proceed.

The qreqn* signals are synchronized internally, and can be driven asynchronously.

As the qactive output signal includes combinatorial and asynchronous inputs, then you must consider qactive as an asynchronous output.

For more information, see the *AMBA® Low Power Interface Specification*.

3.1.5 Distributor P-Channel

The P-Channel is used for power control of the GIC-720AE Distributor.

The P-Channel is present only in multichip configurations. It is used to safely isolate the Distributor from other chips to allow the save and restore of its register states.

Related information

Power management on page 91

3.1.6 Distributor configuration

You can configure several options that relate to the operation of the Distributor block.

Table 3-8: Configurable options for the Distributor

Feature	Range of options
Number of chips	1-64
Affinity level that is used for chip selection.	2, 3
Affinity0 width	0-4
Affinity1 width	0-8
Affinity2 width	0-8
Affinity3 width	0-4
Number of *GIC Cluster Interfaces* (GCIs).	0-256
The GCIs that support real-time interrupts.	0-8
LPI support	True, False
LPI cache depth, or cache entries ÷ 2.	8, 16, 32, 64, 128, 256, 512
Number of LPI cache banks.	1, 2, 4
Number of ITS blocks on the chip.	0-32

Feature	Range of options
Number of credits for transferring LPIs between chips.	1-8
Number of credits for transferring SGIs between chips.	1-8
Number of credits for transferring vSGIs between chips.	1-8
GICv4.1 support	True, False
Number of vPEs supported, $2^{<value>}$.	2-14
Number of standard SPI signals.	32-1984, in blocks of 32. The 1984 SPIs minus any real-time SPIs, can be spread across 32 SPI Collators. To support 1984 SPIs, the cores must support the GICv3.1 extensions, otherwise the maximum is 960 SPIs.
Number of real-time SPI signals.	32-960, in blocks of 32.
Number of SPI Collators.	0-32
Remove cores from a preconfigured GIC.	Options include: • No support for reducing the number of cores. • Secure software can reduce the number of cores. • The gicd_pe_off tie-off signal can reduce the number of cores.
Local chip addressing	• Unified cross-chip addressing. All Distributors use the same addressing scheme. • Local cross-chip addressing. Each Distributor has its own addressing scheme.
RAM I/O support	Enables I/O to be present and routed to each RAM in a subblock. These I/O have no inherent functionality inside the design. You can use the I/O to control elements within your RAM models.
Remove support for 1 of N SPIs.	True, False
Protection for the following AXI5-Stream interfaces: • GICD ↔ GCI • GICD ↔ ITS • GICD ↔ SPI Collator • GICD → Wake Request	Options include: **0** None **1** AMBA parity **2** CRC **3** AMBA parity and CRC
Protection for spi signals.	True, False
Protection for rlt_spi signals.	True, False
MBIST protection	True, False
Q-Channel protection	True, False
P-Channel protection	True, False

Feature	Range of options
APB5 protection	True, False
Protection for input tie-off signals.	True, False
Protection for non-AMBA output signals.	True, False
Protection for PMU sample and request signals.	True, False

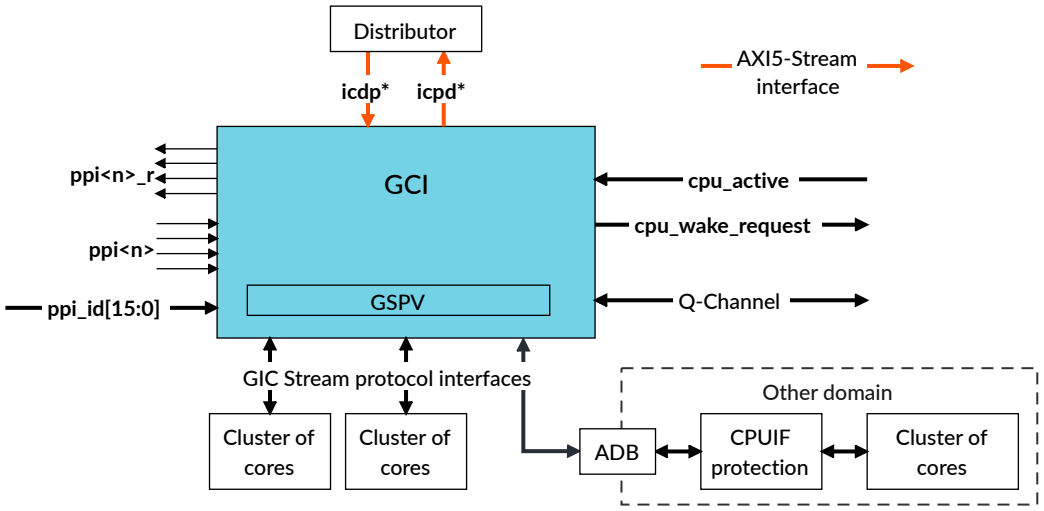
For more information, see the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual*.

3.2 GIC Cluster Interface

The *GIC Cluster Interface* (GCI) is responsible for PPIs and SGIs that are associated with its related cluster or group of cores.

The following figure shows the GCI. The figure does not show the protection signals that a configuration can include.

Figure 3-2: GCI



The GCI performs the following functions:

- Maintaining the SGI and PPI programming.
- Monitoring, and if necessary, synchronizing the PPI wires.
- Prioritizing SGIs, PPIs, and any other interrupts that are sent from the Distributor, and forwarding them to the core. When a GCI is configured to support real-time interrupts, it arbitrates with the Distributor to satisfy the real-time requirements of PPIs and SGIs. At reset, if GICR_FCTLR.ECP ==1, then the GCI supports real-time interrupts.

- Maintaining the GIC Stream protocol and communicating with the cluster. The *GIC Stream Protocol Validator* (GSPV) corrects any errors in the GIC Stream transfers. The GSPV supports mixed criticality systems where some processors might not have an ASIL-D rating.

A *CPU interface* (CPUIF) protection block provides a CRC protected connection from GCI to core, when any of the following apply:

- The GCI is configured to support multiple buses.
- The connection to the core is not a simple point-to-point connection because it contains register slices, an *AMBA Domain Bridge* (ADB), or an unprotected interconnect.
- When the `structure` configuration parameter is set to `wrap`, and the GIC is configured to reduce the number of AXI5-Stream interfaces on the GCI. See 3.8 Hierarchy on page 56 for information about the `structure` parameter.

When using CPUIF protection, the final point-to-point connection can be configured to use AMBA parity or no protection, although AMBA parity is recommended.

There can be multiple GCIs in a configuration and they can be sized to match your system. For example, if you have two clusters of eight cores, then you can have one GCI positioned next to each cluster. You can use a GCI for each cluster to reduce the PPI wiring and enable the GCI to be powered down with the cores for extra power savings. Alternatively, for a small system, combining all cores into one GCI might be the best solution. See *Configuration options* in the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual* for more information.

The GCI (GICR) registers are programmed through the Distributor ACE5-Lite subordinate interface. The Distributor also contains the architectural LPI functionality.

GSPV

The *GIC Stream Protocol Validator* (GSPV) provides a recovery mechanism to recover PEs that do not respond to a GICR_WAKER.ProcessorSleep handshake. In a mixed-criticality system, it should be possible to reset a failing ASIL-B core without impacting the other cores that connect to the GIC. However, to achieve this recovery, it is important to insert the correct clock and reset boundaries to meet these reset requirements. In general, when resetting at the cluster level, the easiest way is to provide a separate GCI for each cluster with a clock and reset boundary between the GCI and GICD, which provides a clean point to reset.

Local PE wake

If a GIC configuration has `ci_wake` == 1, then each GCI has cpu_wake_request signals for the PEs that connect to that GCI. This configuration setting places another set of wakeup signals close to the cores. To wake a PE, the system designer can choose to use the cpu_wake_request signals or the wake_request signals from the Wake Request block.

When a system uses the cpu_wake_request signals, if the system is able to power down a GCI, the system designer must connect the corresponding wake_request signals to a power controller. When the GCI is powered down, the cpu_wake_request signals can not wake the cores, but the use of the wake_request signals enables all cores on that GCI to be woken. See also 3.6 Wake Request on page 53.

Related information

PPIs on page 72

3.2.1 GCI AXI5-Stream interface

Each GCI has an upstream and downstream AXI5-Stream interface for communicating with the Distributor. This interface is 16-bit, 32-bit, or 64-bit wide and uses a fully credited protocol.

3.2.2 GCI GIC Stream Protocol interface

The GIC-720AE uses the GIC Stream Protocol interface to send interrupts to the core and receive notifications when the core activates interrupts.

The GIC Stream Protocol interface has a pair of 16-bit or 32-bit wide AXI5-Stream interfaces, one upstream interface, and one downstream interface. However, if the GCI supports real-time interrupts, then the data width is always 32-bit. At reset, if GICR_FCTLR.ECP ==1, then the GCI supports real-time interrupts.

GIC-720AE uses some extended packets and is designed to work with the Cortex®-R82AE cores. Software can use GICR_FCTLR.ECP to disable these extended packets.

The GIC Stream Protocol interface, also referred to as the GIC Stream interface, uses the GIC Stream protocol to pass interrupts and responses to the CPU interface inside each core.

See the *GIC Stream Protocol interface* appendix in the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.

Table 3-9: GIC Stream Protocol interface signals

Signal	Description
iri<*>	The iri prefix identifies the names of the downstream interface signals. These signals are sent by the GIC Stream transmitter. On this interface, the GCI is the transmitter and the CPU interface is the receiver.
icc<*>	The icc prefix identifies the names of the upstream interface signals. These signals are sent by the GIC Stream transmitter. On this interface, the CPU interface is the transmitter and the GCI is the receiver.
iritdest	The GCI uses this signal to direct packets to one core within the cluster.
icctid	The cluster uses this signal to determine which core within the cluster sent a packet.
iritwakeup	The GCI uses this signal to indicate that it wants to send a message to a CPU interface in the cluster.
icctwakeup	The cluster uses this signal to indicate that it wants to send a message to the GCI.
iri<*>_chk, icc<*>_chk	AMBA parity _chk signals

Both the iritdest and icctid signals can support 64 cores that use packed binary encoding, as opposed to one-hot encoding. They can also be divided down using an AXI5-Stream crossbar to support clusters of an arbitrary number of cores from 1-64.

The necessary crossbar is generated as part of the render process, depending on the number of GIC Stream buses that are specified for each GCI.

3.2.3 GCI Q-Channel

The GCI has a single Q-Channel interface that is used to ensure that the GCI can be safely clock gated hierarchically.

If the GCI is busy, actively processing interrupts or sending messages upstream or downstream, the Q-Channel denies a quiescence request that it receives on the qreqn signal, by asserting the qdeny signal. For more information, see the *AMBA® Low Power Interface Specification*.

The qreqn input signal is synchronized inside the GCI. The qactive signal is connected to the PPI wires directly, and must be considered as an asynchronous output.

If Q-Channels are configured to support AMBA parity, then parity is both generated and checked. Changes in qreqn are not forwarded until it observes consistency for qreqn and qreqn_chk.

3.2.4 GCI PPI signals

GIC-720AE supports 16, 32, or 48 PPIs, and synchronized output return wires, for each core. The number of PPIs and return wires must be the same for all cores that are sharing a GCI.

Level-sensitive PPI signals are active-LOW by default, as with previous Arm GIC implementations. However, individual PPI signals can be inverted and synchronized using the following parameters:

- `FAINLIGHT_<usrcfg>_PPI<ppi_id>_<cpu_number>_<ppi_number>_INV`
- `FAINLIGHT_<usrcfg>_PPI<ppi_id>_<cpu_number>_<ppi_number>_SYNC`
  Where `<usrcfg>` is user-defined text that is assigned when the GIC is configured, which can help with identifying a GIC configuration.

Every ppi<n> signal has a corresponding ppi<n>_r signal from after the synchronizer or capture flop. These ppi<n>_r signals can be used to create pulse extenders for edge-triggered interrupts that cross clock domains. The `FAINLIGHT_<usrcfg>_PPI<ppi_id>_<cpu_number>_<ppi_number>_INV` parameter also inverts the ppi<n>_r signal.

Both ppi<n> and ppi<n>_r have an associated odd parity check signal, that is, ppi<n>_chk and ppi<n>_r_chk.

If you plan to use edge-triggered PPIs and use the Q-Channel to clock gate the GCI hierarchically, then you must include pulse extenders. The pulse extenders ensure that interrupts are not missed while the clock restarts.

For information about the purpose of each PPI used by the core in your system, refer to the relevant core *Technical Reference Manual*.

Related information

PPI signals on page 73
SPI Collator wires on page 50

3.2.5 GCI configuration

You can configure several options that relate to the operation of the GCI.

Table 3-10: Configurable options for the GCI

Feature	Range of options
The number of cores that attach to this GCI.	1-64
The number of PPIs for each core. To support more than 16 PPIs, the core must support the GICv3.1 extensions.	16, 32, 48
Support for real-time PPI and SGI interrupts.	True, False
ECC support for the RAM. See 4.17 Reliability, Accessibility, and Serviceability on page 97 for more information.	True
Data bus width for the GCI processor AXI5-Stream interface.	16, 32 for a standard GCI. 32 for a real-time GCI.
AXI5-Stream data bus width.	16, 32, 64 for a standard GCI. 64 for a real-time GCI.
GIC Stream bus structure.	Flexible buses and domains
Protection for CPU interface signals.	True, False
Protection for cpu_active signals.	True, False
Protection for ppi signals.	True, False

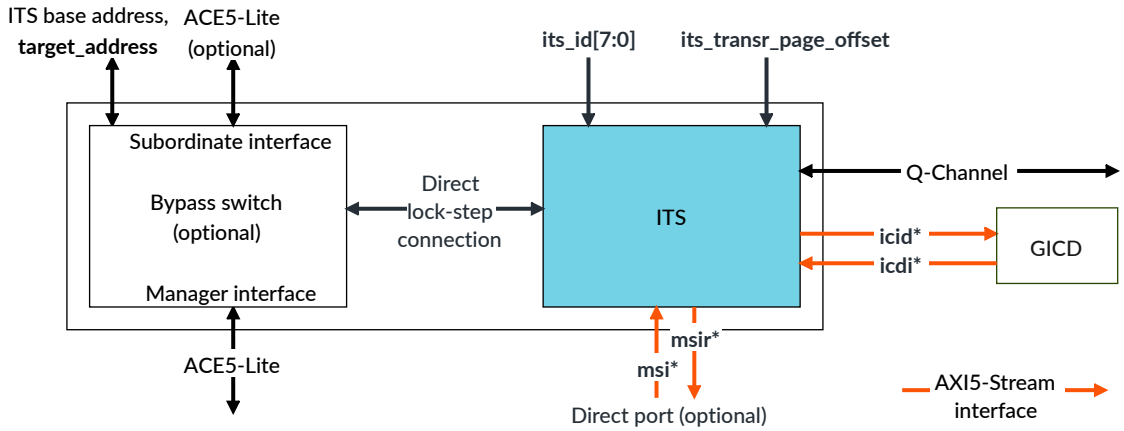
For more information, see the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual*.

3.3 Interrupt Translation Service

The *Interrupt Translation Service* (ITS) provides a software mechanism for translating message-based interrupts into LPIs or vLPIs.

The following figure shows the ITS block, when the GIC is configured to include the optional bypass switch and the optional direct port. The figure does not show the protection signals that a configuration can include.

Figure 3-3: ITS block



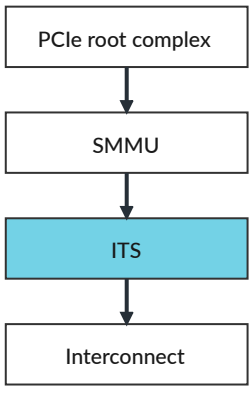
The ITS is an implementation of the GICv3 and GICv4 Interrupt Translation Service as described in the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*. The ITS translates MSI requests to the required LPI and target. It also has a set of commands for managing LPIs for core power management and load balancing.

A main use of the ITS is the translation of MSI/MSIx messages from a PCIe *Root Complex* (RC). To complete the translation, the ITS must be supplied with a DeviceID that is derived from the PCIe RequestorID. To reduce the distance that the DeviceID is transferred and to enable better compartmentalization between RCs, the ITS is best placed next to the RC. To ease integration, the ITS has an optional bypass switch as shown in the ITS block diagram. If the bypass switch is not configured, the ACE5-Lite subordinate port connects to the ITS directly. See 3.3.1 ITS ACE5-Lite subordinate interface on page 43.

See 4.12 ITS on page 82 for more information.

The following figure provides an example of the ITS integration process.

Figure 3-4: ITS integration



An ITS can be placed anywhere in the system so that it is seen by devices that want to send MSIs. However, the system is responsible for ensuring that the DeviceID reaching each ITS, is

not spoofed by rogue software using a<x>user signals or the direct MSI-64 port. See 3.4 MSI-64 Encapsulator on page 47.

If the ITS is placed downstream of an interconnect, care must be taken to avoid system deadlock. For more information, see the *Functional integration guidelines* chapter in the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual*.

3.3.1 ITS ACE5-Lite subordinate interface

The ITS AMBA® ACE5-Lite subordinate interface has a configurable data width of 64 bits, 128 bits, 256 bits, or 512 bits.

The ITS ACE5-Lite subordinate port contains only the GITS_TRANSLATER register. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.

If the bypass switch configuration option is selected, the port accepts all ACE5-Lite traffic, and filters accesses to the ITS based on an address match set by the target_address[ADDR_WIDTH −17:0] ITS base address tie-off. Without the bypass switch, the upper bits of the address, 16 and above, are ignored, and the system address decoders must ensure that only relevant ITS writes arrive at the ITS. Writes to the ITS subordinate interface must set the awaddr[16:0] signal to `0x0040`, irrespective of whether the bypass switch is selected.

The ACE5-Lite subordinate interface ignores all awatop, a<x>snoop, a<x>cache, a<x>domain, and a<x>prot signals information other than to filter non-memory transactions such as atomics and cache maintenance operations, to ensure that it replies in a protocol-compliant manner.

The GIC-720AE uses the wstrb signal to determine the size of a transaction. The GIC rejects transactions where the strobes do not form a continuous block that is address aligned with the resultant size of the transaction.

To generate an LPI, the ITS requires the DeviceID of the issuing manager. For PCIe, the DeviceID is derived from the RequestorID.

The GIC-720AE supports two different methods for deriving the DeviceID with the ACE5-Lite subordinate interface:

- When using the MSI-64 configuration parameter, the write to GITS_TRANSLATER is converted to 64-bit accesses at an unmapped system address and the DeviceID is transferred in the upper 32 bits of the access. In this case, only burst length 1, 64-bit ACE5-Lite writes are accepted.
- When not using MSI-64, the DeviceID is transported on the awuser_s[did_width−1:0] signal during the address (AW) phase of the register access. In this case, burst length 1, 32-bit or 16-bit writes are accepted.

These two modes cannot be mixed on a single ITS. The DeviceID must be transferred using a method that malicious software cannot spoof.

The ITS also supports a direct MSI interface, where MSIs are sent directly on an AXI5-Stream interface to the ITS. See 3.3.3 MSI delivery interface on page 46. This interface can be configured alongside or instead of an ACE5-Lite subordinate interface.

If the bypass switch is configured, it includes a transaction tracker that ensures PCIe ordering requirements are met. The transaction tracker allows continuous downstream traffic including interleaved MSIs, unless the buffer slots become full. There are two buffers, `bypass_max_outstanding`, which specifies the number of concurrent downstream transactions allowed and `bypass_interrupt_count`, which specifies the number of concurrent MSIs that can be waiting for their prerequisite transactions to complete.


Note

- The ITS subordinate port contains only write-only registers, so the read channel always uses a simple transaction tracker that only allows transactions to one destination at a time.
- If the Distributor and ITS both share the ACE5-Lite subordinate port, the port properties match those of the Distributor ACE5-Lite subordinate port, which 3.1.2 Distributor ACE5-Lite subordinate interface on page 29 describes.

The following table shows the acceptance capabilities of the ITS ACE5-Lite subordinate interface.

Table 3-11: ITS ACE5-Lite subordinate interface acceptance capabilities

Attribute	With bypass switch	Without bypass switch
Combined acceptance capability	Read acceptance capability + Write acceptance capability	3
Read acceptance capability	512	1
Read data reorder depth	512	1
Write acceptance capability	`bypass_max_outstanding`, but not exceeding 256	2

The ITS ACE5-Lite subordinate interface has an associated awakeup signal. To ensure that incoming traffic wakes the ITS correctly when it is clock gated hierarchically through the Q-Channel, the awakeup signal must be driven from a registered version of the awvalid and arvalid signals. To prevent spurious wake events, ensure that the awakeup signal is registered cleanly.

3.3.1.1 AMBA bus properties, ITS

The AMBA® protocols define multiple property types that indicate the capabilities of a device.

The following table lists the ACE5-Lite properties of an ITS.

Table 3-12: GIC-720AE ITS ACE5-Lite subordinate interface properties

AMBA property	ITS subordinate interface	PCIe forwarding	ACE5-Lite issue
Atomic_Transactions	Ignore and respond legally	Forwarded	F
Barrier_Transactions	False	False	F

AMBA property	ITS subordinate interface	PCIe forwarding	ACE5-Lite issue
Cache_Stash_Transactions	Basic when `axi_cache_stashing_support` == 0. Ignore and respond legally. True when `axi_cache_stashing_support` == 1. Dataless cache stash operations are supported.	Support non-dataless, forwarded	F
Check_Type	False, Odd_Parity_Byte_All	False, Odd_Parity_Byte_All	F
CMO_On_Read	Ignore and respond legally	Forwarded	G
CMO_On_Write	False	False	G
Coherency_Connection_Signals	False	False	F
DeAllocation_Transactions	Ignore and respond legally	Forwarded	F
DVM_v8	False	False	F
DVM_v8.1	False	False	F
DVM_v8.4	False	False	H
DVM_v9.2	False	False	J
Exclusive_Accesses	False	True	F
InvalidateHint_Transaction	Ignore and respond legally	Forwarded	J
Loopback_Signals	True	Forwarded	F
Max_Transaction_Bytes	4096	4096	F
MPAM_Support	False	Forwarded	G
MTE_Support	Ignore and respond legally	Forwarded	H
NSAccess_Identifiers	False	False	F
Persist_CMO	Ignore and respond legally	Forwarded	F
Poison	Logged or reported but otherwise ignored on reads	Forwarded	F
Prefetch_Transaction	False	False	H
QoS_Accept	False	False	F
Read_Data_Chunking	True	True	G
Read_Interleaving_Disabled	No read data interleaving	Read data interleaving is accepted	G
RME_Support	Ignore and respond legally	Forwarded	J
Shareable_Transactions	True	True	F
Trace_Signals	True	Forwarded	F
Unique_ID_Support	True	Forwarded	G
Untranslated_Transactions	False	False	F
Wakeup_Signals	True	True	F
Write_Plus_CMO	False	False	H
WriteEvict_Transaction	True	Forwarded	F

3.3.2 ITS AXI5-Stream interface

The ITS AXI5-Stream interface is a bi-directional interface of either 16-bit or 64-bit width, for communication between the ITS and the Distributor on the same chip.

We expect that a typical distributed system is 16 bits wide. When a pre-existing wide interconnect is used, the 64-bit option allows messages to be efficiently packed.

The interface is fully credited so all messages can be accepted without dependency on any other ports.

3.3.3 MSI delivery interface

The MSI delivery interface is a bidirectional AXI5-Stream interface for passing MSIs to an ITS for translation.

The data format on the msitdata signal is {DeviceID[31:0], EventID[31:0]}.

When the ITS accepts the request, it sets the msirtvalid signal HIGH.

The GIC decodes the entire 32 bits of DeviceID and EventID. Bits above the configured widths must be zero, otherwise the GIC generates out-of-range errors and the expected translation does not occur.

The msitid signal value that the ITS receives, is sent out on the msirtdest signal. This behavior enables multiple sources to connect to the ITS using a standard AXI5-Stream infrastructure.

The MSI delivery interface can apply back pressure if the ITS or Distributor resources become busy, and can be dependent on the Distributor ACE-Lite manager interface, for both reads and writes.

3.3.4 ITS Q-Channel

The ITS has a Q-Channel interface which controls requests from an external clock gating source.

If the ITS is busy, the Q-Channel interface asserts the qdeny signal to deny an external request to gate its clock. When an external request occurs, the interface requests a wakeup by asserting the qactive signal.

The qreqn input signal is synchronized to the ITS.

3.3.5 ITS configuration

You can configure several options that relate to the operation of the ITS block.

Table 3-13: Configurable options for the ITS

Feature	Range of options
DeviceID width	3-24
EventID width	1-20

Feature	Range of options
CollectionID width	2-14
Inclusion of a bypass port.	True, False
MSI-64 support, which controls whether the DeviceID is sent using the awuser signals or on bits[63:32] that are written to GITS_TRANSLATER. See 4.12.2 MSI-64 on page 84.	True, False
Include an ACE5-Lite subordinate interface for writes to GITS_TRANSLATER (or for bypass).	True, False
Include an AXI5-Stream port for transferring “writes” to GITS_TRANSLATER. Other devices can use this port to avoid using address-mapped transactions.	True, False
The number of credits for supporting transfer of LPIs using locked translations to the Distributor.	0-4
The number of credits for supporting transfer of LPIs using non-locked translations to the Distributor.	1-16
ACE5-Lite subordinate interface address width.	20-52
ACE5-Lite subordinate interface data width.	64, 128, 256, 512
ACE5-Lite subordinate interface read ID width.	1-32
ACE5-Lite subordinate interface write ID width.	1-32
ACE5-Lite loop signal width.	1-8
AXI5-Stream data width	16, 64
ECC support for the caches. For more information, see 4.17 Reliability, Accessibility, and Serviceability on page 97.	True
Collection cache depth, or cache entries ÷ 2.	2, 4, 8, 16, 32, 64, 128, 256, 512
Device cache depth, or cache entries ÷ 2.	2, 4, 8, 16, 32, 64, 128, 256, 512
Event cache depth, or cache entries ÷ 2. The number of Device and EventID pairs that an ITS caches.	2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048
Domain name. For more information, see Figure 3-9: GIC top-level structure options on page 57.	Any legal domain identifier
AMBA parity protection for the optional direct ports.	True, False
AMBA parity protection for all ITS ACE5-Lite interfaces.	True, False

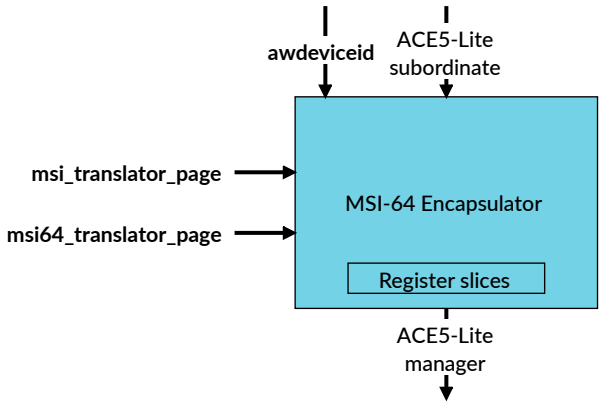
For more information, see the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual*.

3.4 MSI-64 Encapsulator

The MSI-64 Encapsulator reduces system wiring by combining the DeviceID onto the data bus for writes to the GITS_TRANSLATER register.

The following figure shows an overview of the MSI-64 Encapsulator process. The figure does not show the protection signals that a configuration can include.

Figure 3-5: MSI-64 Encapsulator



The MSI-64 Encapsulator detects translations that target the page address of the GITS_TRANSLATER register, which is set by the msi_translator_page tie-off signal. It then converts accesses to 64-bit writes, with the awdeviceid signal value in the upper 32 bits of the data and retargets them to the msi64_translator_page signal. This avoids having to use wires to transfer a DeviceID to the GITS_TRANSLATER register for translation.

See 4.12.2 MSI-64 on page 84 for more information.

The silicon rendering process produces two versions of the MSI-64. One version includes the _chk signals and the other version excludes the _chk signals.

3.4.1 MSI-64 ACE5-Lite interfaces

The MSI-64 Encapsulator has an ACE5-Lite subordinate interface and an ACE5-Lite manager interface.

MSI-64 ACE5-Lite subordinate interface with awdeviceid

This interface is a full ACE5-Lite subordinate port with an extra awdeviceid input signal, which is valid, and must remain stable with the awvalid signal.

MSI-64 ACE5-Lite manager interface

This interface is a full ACE5-Lite manager port.

The following table shows the transaction acceptance capabilities of both subordinate and manager ports.

Table 3-14: Transaction acceptance

Transaction type	Maximum number of transactions allowed
Read	Unlimited
Write	Unlimited
Combined	Unlimited

Any leading wdata signal is registered and held until the awaddr signal arrives.

Note

- The MSI-64 Encapsulator requires a data bus that has a width of 64 bits or greater.
- The ACE5-Lite manager port never issues more than two addresses before the wlast signal asserts.
- CMOs that target the addresses that the msi_translator_page signal selects, are converted to single beat reads.

3.4.2 MSI-64 Encapsulator configuration

The MSI-64 Encapsulator does not have any configurable parameters at design time. However, if this block is generated in your RTL design, it has several options that you can configure at build time.

The MSI-64 Encapsulator is generated as part of any GIC configuration that includes an MSI-64 enabled ITS.

The following table shows the options for the MSI-64 Encapsulator that you can configure at build time.

Table 3-15: Configurable options for the MSI-64 Encapsulator

Build-time option	Function	Range of options
DATA_WIDTH	Specifies the width of rdata and wdata data signals.	64, 128, 256, 512
ADDR_WIDTH	Specifies the width of araddr and awaddr address signals.	17-52
AWUSER_WIDTH	Specifies the width of awuser signal.	1-128
ARUSER_WIDTH	Specifies the width of aruser signal.	1-128
RUSER_WIDTH	Specifies the width of ruser signal.	1-128
WUSER_WIDTH	Specifies the width of wuser signal.	1-128
BUSER_WIDTH	Specifies the width of buser signal.	1-128
DID_WIDTH	Specifies the width of the DeviceID.	3-24
WID_WIDTH	Specifies the width of wid signal.	1-32
RID_WIDTH	Specifies the width of rid signal.	1-32
ARLOOP_WIDTH	Width of the arloop and rloop signals.	1-8
AWLOOP_WIDTH	Width of the awloop and bloop signals.	1-8
AWDEVICEID_FROM_AWUSER	Extract Device ID from awuser signals.	0, 1
AWUSER_AWDEVICEID_BASE	Base of Device ID in awuser signals. Used when AWDEVICEID_FROM_AWUSER == 1.	0-125
FWD_REG_TYPE	Register slice type on forward AW, AR, and W channels.	**0** None **1** Reverse **2** Forward **3** Full

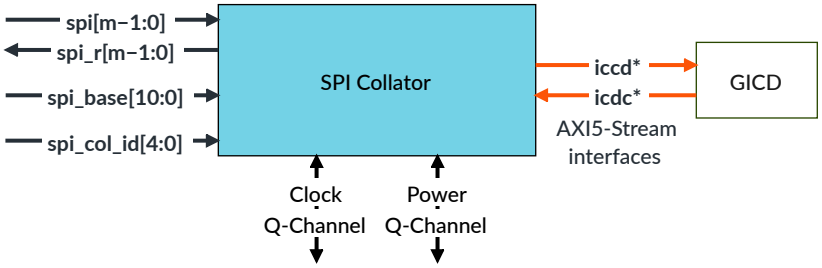
Build-time option	Function	Range of options
REV_REG_TYPE	Register slice type on B and R channels.	**0** None **1** Reverse **2** Forward **3** Full

3.5 SPI Collator

The SPI Collator converts SPI wires into messages to be sent to the Distributor. The GIC can be configured to provide up to 32 SPI Collators.

The following figure shows an SPI Collator block. The figure does not show the protection signals that a configuration can include.

Figure 3-6: SPI Collator



Individual SPIs can be synchronized into the SPI Collator, or an SPI Collator can be placed in the same clock domain as the interrupt sources and the messages that are synchronized into the Distributor.

Placing the SPI Collators in clock domains that are always on and remote from the GIC Distributor, enables more aggressive power saving because the Distributor can be clock gated hierarchically.

3.5.1 SPI Collator AXI5-Stream interface

The AXI5-Stream interface enables communication between an SPI Collator and the Distributor.

The AXI5-Stream ports apply only transient backpressure to the AXI5-Stream interface, which enables packets to be routed over any free-flowing interconnect.

3.5.2 SPI Collator wires

The SPI Collator wires can be extended to create other functions.

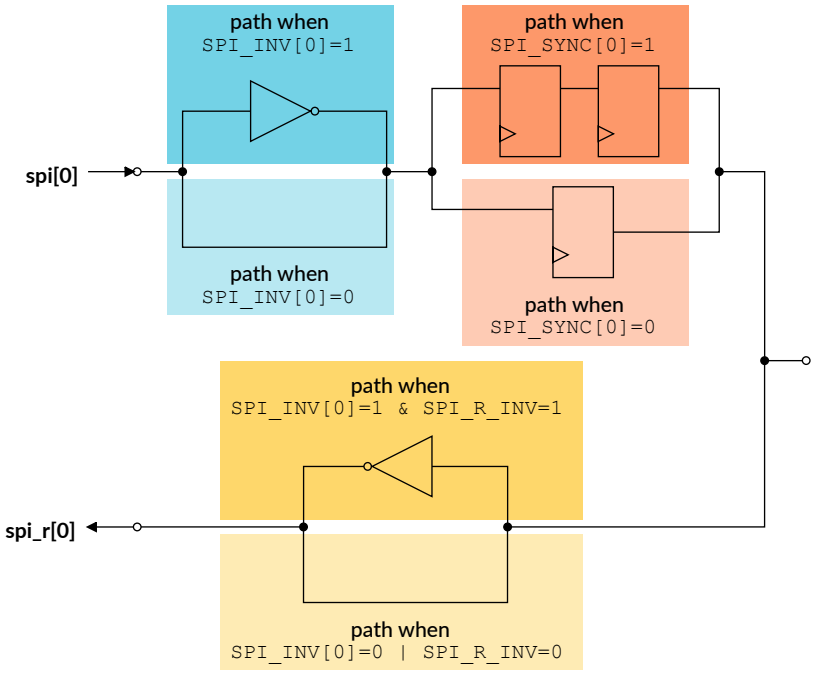
By default, the asserted level of an SPI is active-HIGH, as with previous Arm GIC implementations. However, each SPI can be either inverted, synchronized, or both, using the `SPI_INV[n]` and `SPI_SYNC[n]` build-time options, where:

- SPI_INV[n] == 1 indicates that the inverter is enabled.
- SPI_SYNC[n] == 1 indicates that the synchronizer is enabled.
- [n] = SPI_ID – 32.

Each SPI input wire, spi, has a corresponding spi_r wire after the synchronizer or capture flop that can be used to create interrupt pulse extension for edge-triggered interrupts that cross clock domains. If SPI_INV[n] is set to 1, then the wire after the synchronizer is inverted with respect to the input unless the SPI_R_INV option is set to 1. If the SPI_R_INV option is set to 1, then it removes any inversion that SPI_INV[n] applies to individual SPIs on that SPI Collator.

The following figure shows the effect of the SPI_INV[0], SPI_SYNC[0], and SPI_R_INV build-time options on the spi[0] signal.

Figure 3-7: SPI parameters and signal conditioning



3.5.3 Using multiple SPI Collators

If a GIC configuration uses multiple SPI Collators, then the SPI_BASE value must be set so that the SPI wires do not overlap.

The SPI_BASE value controls the base address of an SPI Collator, and it is set by using either an SPI_BASE build-time option or an spi_base signal. The choice of whether to use build-time options or signals, to set the base address of all SPI Collators on the chip, is decided during configuration.

For example, if the chip uses the SPI_BASE build-time option to set the base addresses of its three SPI Collators, then the SPI_BASE options could be set to:

- 1 SPI Collator with 64 wires - SPI_BASE 0
- 1 SPI Collator with 32 wires - SPI_BASE 64
- 1 SPI Collator with 128 wires - SPI_BASE 96

SPI Collators do not have to support a multiple of 32 wires.

3.5.4 SPI Collator power Q-Channel

The SPI Collator has a power Q-Channel interface that accepts requests from an external source, such as the system power controller.

When the qactive_col signal is LOW, it indicates that all SPIs to the SPI Collator are in their idle state of either 0 (active-HIGH) or 1 (active-LOW), so all messages are sent to the Distributor.

If the qactive_col signal is HIGH, the SPI Collator rejects any attempt to enter a low-power mode.

If the qreqn_col signal is LOW and is accepted, the SPI Collator enters low-power mode and the AXI5-Stream channels to the Distributor are flushed out to ensure that there are no messages in progress. When accepted, you can reset the SPI Collator safely without having to also reset the Distributor. You can also reset the Distributor, but you must first complete the instructions that are described in the subsections of section 4.15 Power management on page 91 before the Distributor can be powered down.

When the SPI Collator and Distributor are both in the same domain, the power Q-Channel interface is redundant and can be tied off.

In low-power mode, it is only safe to stop the SPI Collator clock if all edge-triggered interrupts into the SPI Collator are pulse extended so that edges are not missed.

3.5.5 SPI Collator clock Q-Channel

The SPI Collator has a clock Q-Channel interface that accepts requests from an external clock gating source, such as the system clock controller.

When the qactive_col_clk signal is LOW, it indicates that all SPI toggles and level transitions have been passed to the Distributor, and that the SPI Collator does not require the clock.

If the qactive_col_clk signal is HIGH, the SPI Collator rejects any attempt to enter a low-power mode.

If the qreqn_col_clk signal is LOW and is accepted, the SPI Collator enters low-power mode and no new messages are sent to the Distributor until it enters low-power mode. If any interrupt line changes state, the qactive_col_clk signal is asserted.

In low-power mode, it is only safe to stop the SPI Collator clock if all edge-triggered interrupts into the SPI Collator are pulse extended so that edges are not missed.

3.5.6 SPI Collator configuration

You can configure several options that relate to the operation of an SPI Collator block.

Table 3-16: Configurable options for an SPI Collator

Feature	Range of options
The number of standard SPI wires. The total number of SPIs on all SPI Collators must be ≤1984 minus <number of real-time SPIs>.	1-1024, in multiples of 32.
The number of SPI Collators.	0-32
SPI_INV is a wide vector of one bit for each SPI, indicating whether to invert the interrupt. This parameter is a build-time option.	True, False
SPI_SYNC is a wide vector of one bit for each SPI, indicating whether to synchronize the interrupt. This parameter is a build-time option.	True, False
SPI_R_INV is a single bit, indicating whether to invert the return path for any spi_r signals where SPI_INV[n] == 1. This parameter is a build-time option. See 3.5.2 SPI Collator wires on page 50.	True, False
Base address tie-off signal support.	**0** The `SPI_BASE` build-time option sets the ID of the starting SPI_ID for this SPI Collator. `SPI_BASE` can be set to 0-1983. **1** The spi_base[10:0] signal sets the ID of the starting SPI_ID for this SPI Collator.
SPI_PROT_RESET_DISABLED is a wide vector of one bit for each SPI, indicating whether to disable the interrupt protection for that SPI. The GIC detects the settings as it exits reset. This parameter is a build-time option.	True, False
SPI_PROT_RESET_PERMONLY is a wide vector of one bit for each SPI, indicating whether the interrupt protection detects permanent faults only for that SPI. The GIC detects the settings as it exits reset. If False, the SPI protection detects permanent and transient faults. This parameter is a build-time option.	True, False

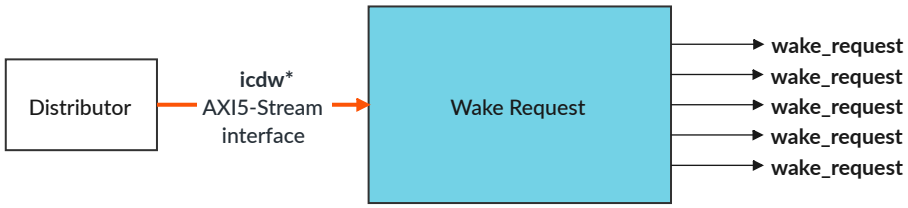
For more information, see the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual*.

3.6 Wake Request

The Wake Request block converts AXI5-Stream wake requests into one wake_request signal for each core. Each wake_request signal connects to the system power controller.

The following figure shows the Wake Request block. The figure does not show the protection signals that a configuration can include.

Figure 3-8: Wake Request



A wake_request signal wakes a powered-down core when one of the following conditions is true:

- An interrupt that targets only that specific core is pending.
- GICD_CTLR.E1NWF is set, and a 1 of N SPI selects that core as its target.

The GIC-720AE does not know whether a core is powered up or down. It only knows whether software has enabled sending transactions on the AXI5-Stream interface. Therefore, a wake_request signal remains asserted after a core has powered up. A wake_request signal deasserts when software clears GICR_WAKER.ProcessorSleep and the GIC-720AE clears the GICR_WAKER.ChildrenAsleep bit.

If there are pending interrupts, either targeted or 1 of N, when GICR_WAKER.ProcessorSleep is set, the wake_request signal might assert during the powerdown sequence. The power controller must ignore the wake_request signal until the core is powered down.

An asserted wake_request[<cpus>−1:0] signal deasserts only when:

- The Distributor exits reset, which causes it to send a clear message to the Wake Request block.
- The core is woken and software clears the GICR_WAKER.ProcessorSleep bit, which indicates that the core is able to communicate with the GIC.
- The Wake Request block is reset. If the system resets the Wake Request block, then it must also reset the Distributor.

Core removal support

If a GIC configuration supports the removal of cores, then it is possible to modify how the GIC drives the wake_request bus. The `wake_compress` configuration parameter controls how the bus is driven as follows:

`wake_compress` == 0

The GIC drives the wake_request bus by using a fixed mapping between a core and its corresponding wake_request signal. Use this setting when each core has its own power control logic.

`wake_compress` == 1

The GIC only uses the lower bits of the wake_request bus when either Secure software or the gicd_pe_off[max_pe_on_chip − 1:0] signal removes some cores from the configuration. For example, if a configuration supports 16 cores and software or hardware removes 12 cores, then the GIC only uses the wake_request[3:0] signals. Use this setting when a centralized processor controls the power logic of the cores that remain.

See A.1 Removing cores from a preconfigured GIC on page 361 for more information.

3.6.1 Wake Request AXI5-Stream interface

The AXI5-Stream interface enables the Wake Request block to communicate with the Distributor.

The AXI5-Stream interface does not exert back-pressure.

3.6.2 Wake Request configuration

The Wake Request block has a single configuration option.

Table 3-17: Configurable options for Wake Request

Feature	Range of options
Compress the width of the wake_request[<cpus> – 1:0] signal	0, 1

3.7 Interconnect

The GIC-720AE uses AXI5-Stream interfaces for communication between some blocks.

These blocks are:

- Distributor to, and from, ITS
- Distributor to, and from, Redistributors
- Distributor to Distributor for cross-chip communications
- Distributor to, and from, an SPI Collator
- Distributor to the Wake Request block

All these interfaces use fully credited schemes where all messages are guaranteed to be accepted without dependency on any other port.

Apart from the cross-chip communications, GIC-720AE provides an AXI5-Stream interconnect for transporting messages. However, messages can be sent over an existing interconnect provided the interconnect is free-flowing.

3.7.1 Interconnect configuration

The internal interconnect is configured automatically in accordance with the number of cores and ITS blocks in the system. The configuration produces a balanced tree structure with minimum *Clock Domain Crossings* (CDCs).

The Arm internal scripts limit a single interconnect crossbar to 16 destinations. To work around this limitation, you can use domains in the config file. For example, instead of 32 GCIs in one domain, you can use two domains that each contain 16 GCIs.

3.8 Hierarchy

The hierarchy of the GIC components can be selected using the `structure` configuration parameter.

The `structure` configuration parameter has the following options:

wrap This option provides the lowest level of structure, and wraps the following blocks:

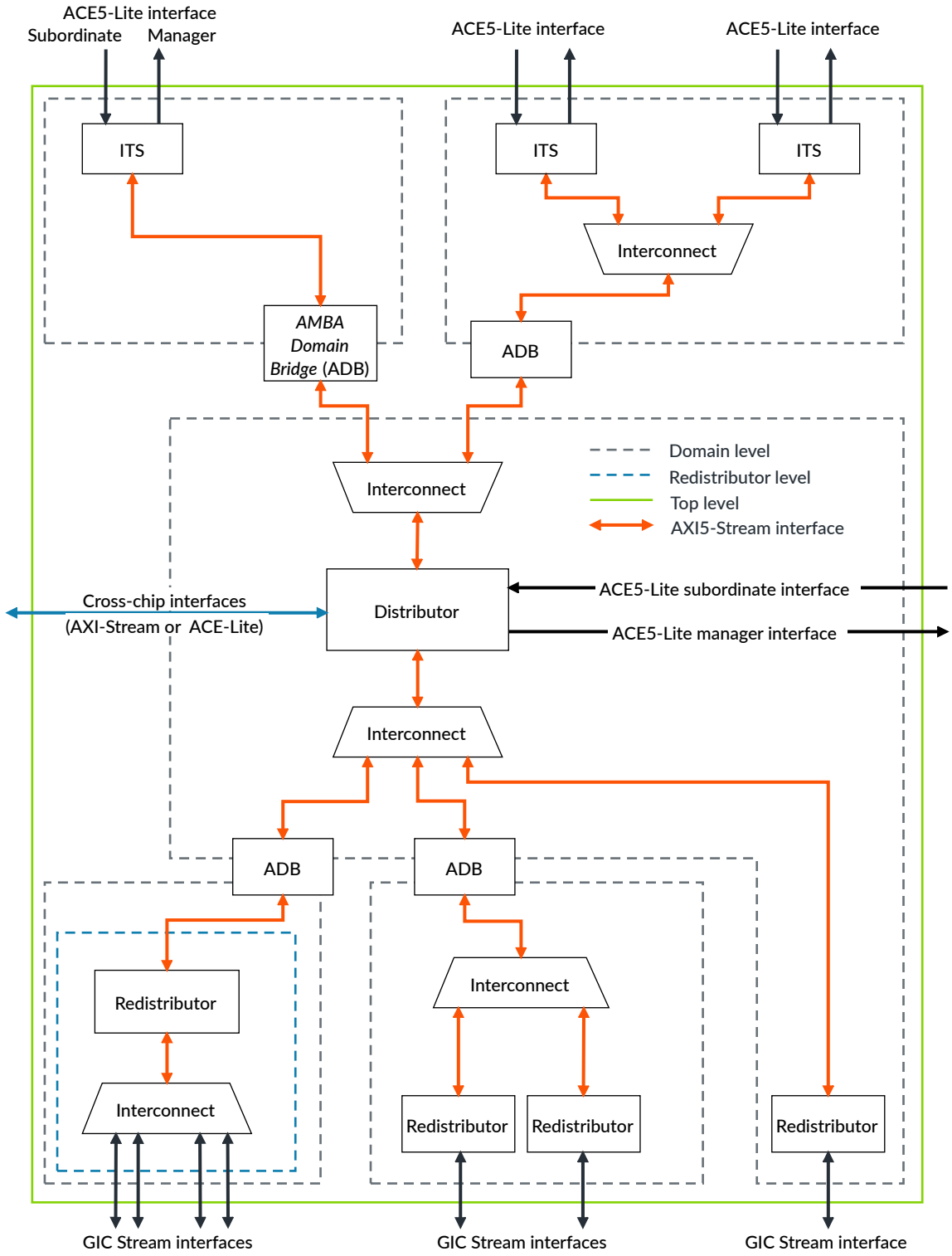
- The Redistributor is wrapped with interconnect components between the Redistributor and the cores. The components that are wrapped at this level are shown within the blue dashed lines in the following figure. If the core is in a different clock domain, in accordance with the domain tags, then half of the CoreLink™ ADB-400 domain bridge is included in the `fainlight_ppi_wrap_<n>_<usrcfg>.v` stitched file.
- The ITS is wrapped (along with any selected bypass switch) in the `fainlight_its_wrap_<n>_<usrcfg>.v` file.
- The GICD is wrapped, including an ITS if the `monolithic` parameter is set to 1, in the `fainlight_gicd_wrap_<usrcfg>.v` file.

domain All blocks and wrapped components that are in the same domain are stitched together in a file that is called `fainlight_domain_<name>_<usrcfg>.v` and includes ADB-400 domain bridges and collated low-power interfaces. Blocks and components at this level are shown within the red dashed lines in the following figure.

full All domains are stitched together to create a single top-level GIC-720AE file, `fainlight_<usrcfg>.v`.

The following figure shows the top-level options.

Figure 3-9: GIC top-level structure options



4. Operation of GIC-720AE

This chapter provides an operational description of the GIC-720AE product.

4.1 Interrupt types

The GIC-720AE manages real-time SPIs, SPIs, SGIs, PPIs, and LPIs. When GICv4.1 is enabled, SGIs and LPIs can be injected into *virtual Processing Elements* (vPEs).

4.2 Multichip operation

During silicon configuration, the system designers can configure the GIC-720AE to support multichip operation.

Systems that comprise more than one chip, can have several SoCs that are connected externally or an SoC comprising several SoCs connected inside a single physical package. In all cases, each SoC is integrated with a GIC-720AE. A multichip system can have up to 16 chips.

To control the consistency of all chips in the configuration, and make the GIC appear as a single entity to the OS, the GIC-720AE uses a set of registers that define the connectivity between chips. These registers are referred to as the Routing table and consist of the following three register types:

GICD_CHIPR<n>
: These Chip Registers define the Routing table. It specifies the SPIs that the chip owns, and how the chip is accessed. This register exists on each chip in the multichip configuration so that each chip has a copy of the Routing table. The register number <n> corresponds to the value of its chip_id signal.

GICD_DCHIPR
: The Default Chip Register specifies the current chip that is responsible for the consistency of the Routing table, and indicates when an update is in progress. A single copy of this register exists on each chip in the multichip configuration.

GICD_CHIPSR
: The Chip Status Register specifies details of the current status of the chip. A single copy of this register exists on each chip in the multichip configuration.

At reset, each chip in the multichip system configuration is effectively a standalone full-featured GIC. The GICD_CHIPSR register on the chip indicates this state with bit RTS == Disconnected.

For the multichip configuration to be fully coherent, all chips in the configuration must be interconnected and one chip must own the Routing table.

The sequence for connecting chips together is described in A.5 Connecting the chips on page 367.

When multiple chips in the configuration are connected, each set of 32 SPIs (SPI block) is owned by a specific chip, so that the SPI space between chips is partitioned. Also:

- SPIs that are not owned by any chip in accordance with the Routing table cannot be used.
- SPI wires on a chip can only be used for SPIs that are owned. However, message-based accesses to SPIs owned on any chip are supported.
- The Routing table can only process one operation at a time. Therefore, software must ensure that GICD_DCHIPR.PUP == 0 before commencing any operation such as writes to GICD_CHIPRx or GICD_DCHIPR.

Local cross-chip addressing

The GIC provides the `local_chip_addr` configuration parameter that controls whether all chips use the same address to reach a destination chip or each chip has its own local addressing to another chip.

When `local_chip_addr` = 0, all chips use the same address to reach a destination chip, so the addressing to a given chip must be the same from each start point to that chip.

When `local_chip_addr` = 1, each chip has its own local addressing to the other chips, which can differ between chips. This setting enables addressing where the address for a fixed chip endpoint can be different between the startpoints.

Software can discover the state of `local_chip_addr` by reading the GICD_CFGID.LCA bit.

4.3 Interrupt groups and security

The GIC-720AE configures the interrupts that it receives into one of three groups. Each group determines the security status of an interrupt and how it is routed.

The following registers control to what group each interrupt is assigned:

- GICD_IGROUPRn and GICD_IGROUPRnE
- GICD_IGRPMODRn and GICD_IGRPMODRnE
- GICR_IGROUPR0 and GICR_IGROUPR1E
- GICR_IGRPMODR0 and GICR_IGRPMODR1E

The groups are:

- Group 0
- Group 1 Secure
- Group 1 Non-secure

Each interrupt is programmed to belong to an interrupt group. Each interrupt group:

- Determines the Security state for interrupts in that group, depending on the Exception level of the core.
- Has separate enable bits that control whether interrupts in that group can be forwarded to the core.
- Has an impact on later routing decisions in the core interfaces.

The GIC-720AE supports the three interrupt groups that the following table shows.

Table 4-1: Security and groupings

Interrupt type	Example use
Secure Group 0	Interrupts for EL3 (Secure firmware).
Secure Group 1	Interrupts for Secure EL1 (Trusted OS).
Non-secure Group 1	Interrupts for the Non-secure state (OS and the hypervisor, or one of both).

The following table shows the interrupt signals that are used for each interrupt group, Security state, and Exception level.

Table 4-2: Interrupt signals, Security states, and Exception levels

Core Exception level and Security state	Group 0	Group 1	
		Secure	Non-secure
Secure EL0, EL1	FIQ	IRQ	FIQ
Non-secure EL0, EL1, EL2	FIQ	FIQ	IRQ
EL3	FIQ	FIQ	FIQ

When the GIC exits reset, the gicd_ctlr_ds tie-off signal controls the GIC-720AE security as follows:

gicd_ctlr_ds and gicd_ctlr_ds_chk are LOW
: Security enabled

gicd_ctlr_ds and gicd_ctlr_ds_chk are HIGH
: Security disabled

Setting the gicd_ctlr_ds tie-off signal HIGH removes the security support of the GIC-720AE. Software can determine the state of this signal by reading the GICD_CTLR.DS bit. When the system has no concept of security, the gicd_ctlr_ds signal must be set HIGH to allow access to important registers.

If the gicd_ctlr_ds signal is HIGH, only a single Security state is supported. In a single Security state, register access, and the behavior and number of interrupt groups supported are affected. For more information, see *Interrupt grouping* and *Interrupt grouping and security* in the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.


Note

We recommend that the gicd_ctlr_ds signal is only set HIGH when your system does not support security.

Group 0 is always Secure in systems with security. If you decide to write security-unaware software using Group 0, it might not be portable to systems with a concept of security. Security-unaware software is most portable when written using Group 1.

If a system has a concept of security but one or more cores do not, then you must not disable security. Instead each core is only able to enable the interrupt groups corresponding to the Security states that it supports.

If you know that your system is always security aware, then we recommend setting the gicd_ctlr_ds signal LOW.

For more information, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* and the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4*.

Security for multi view configurations

A GIC configuration that supports multi view, provides four logical views so that up to three different OSs within a system can be assigned a different view. The GIC supports multi view when GICD_CFGID.VIEW == 1.

The GIC provides a gicd_ctlr_ds[3:0] signal so that each view can be assigned to the Secure or Non-secure state. However, if gicd_ctlr_ds[0] is set HIGH, which sets view 0 to a single security state, then gicd_ctlr_ds[3:1] must also be set HIGH.

4.4 Affinity routing and assignment

The GIC-720AE uses affinity routing, a hierarchical scheme, to identify connected cores and for routing interrupts to specific cores.

The Arm architecture defines a register in a core that identifies the logical address of the core in the system. This register, which is known as the *Multiprocessor Identification Register* (MPIDR), has a hierarchical format. Each level of the hierarchy is known as an affinity level, with the highest affinity level specified first:

- For 32-bit Armv8 processors, the MPIDR defines three levels of affinity, with an implicit affinity level 3 value of 0.
- For 64-bit Armv8 processors, the MPIDR defines four levels of affinity.

The GIC-720AE regards each hardware thread of a processor that supports multiple hardware threads as a single independent core.

The affinity of a core is represented by four 8-bit fields using dot-decimal notation, <Aff3>.<Aff2>.<Aff1>.<Aff0>, where Affn is a value for affinity level `n`. An example of an identification for a specific core would be 0.255.0.15.

The affinity scheme matches the format of the MPIDR_EL1 register in Armv8-A. System designers must ensure that the ID reported by the core of the MPIDR_EL1 register matches how the core is connected to the interrupt controller.

The GIC-720AE allows fully flexible allocation of MPIDR. However, it has two built-in default assignments that are based on the `aff0_thread` configuration parameter:

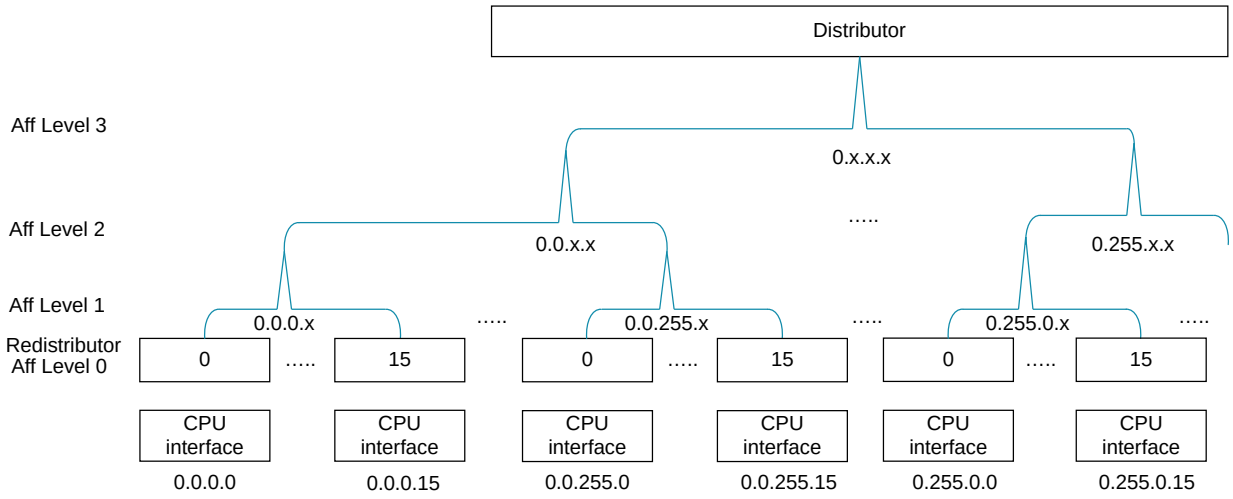
`aff0_thread` == 1
The four fields map to 0.<cluster>.<core>.<thread>

`aff0_thread` == 0
The four fields map to 0.0.<cluster>.<core>

See the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual* for information about the `aff0_thread` configuration parameter and how to build affinity schemes that include heterogenous clusters and multithreaded cores.

The following figure shows the affinity hierarchical structure.

Figure 4-1: Affinity routing



The GIC-720AE can support up to 16 nodes at level 3, with each node able to host 256 child level 2 nodes. Similarly each level 2 node can host 256 level 1 nodes. However, level 1 nodes can only host 16 child level 0 nodes.

If you enable the core removal functionality, then it alters how the MPIDR values are assigned to each Redistributor. See A.1 Removing cores from a preconfigured GIC on page 361 for more information.

For more information about affinity routing, see the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4* and the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

4.5 Multi view

The multi view feature allows software to allocate GIC resources into three or fewer different views. This feature allows control firmware to allocate the GIC to three, or fewer, different OS or hypervisors that are running independent software stacks.

Each view operates independently. There are no behavioral dependencies between the views.

The `multi_view_support` parameter controls whether a GIC configuration supports multi view. Software can read GICD_CFGID.VIEW to check if multi view is supported. In a multichip system, the value of `multi_view_support` must be consistent across all chips.

When this feature is configured, all PEs reset into view 0, which is the configuration view. The other views are view 1, view 2, and view 3. All accesses to view 0, must be a Secure access.

The firmware must assign all PEs to a nonzero view using their corresponding GICR_VIEWR register. In configurations that support programmable removal of PEs, that is GICD_CFGID.RDC == 1, then firmware must:

1. Program the GICD_RDOFFRn register, if software removes some cores.
2. Program the GICR_MPIDR registers, if changes to the affinity values are necessary.
3. Program the GICR_VIEWR registers, if GICD_CFGID.VIEW == 1.
4. Program any other registers.
   Software must not change the GICD_RDOFFRn, GICR_MPIDR, and GICR_VIEWR registers, unless a GIC reset occurs.

The view of each register access is controlled by the 2 LSBs that are above the register page bits of the GIC.

GIC operation in multi view

Each view has a separate gicd_ctlr_ds tie-off signal that controls whether that view operates with a single Security state or two Security states. However, if view 0 operates in a single Security state, then all views must operate in a single Security state.

The GICD_CTLR register is banked by view. The view 0 Group enable controls, that is the EnableGrp* bits, are an overall mask to the Group enables of the other views. Any power state changes or multichip connection changes, require software to clear the view 0 Group enables.

The GICR_SGIDR register is banked by view. Each PE must use the GICR_SGIDR that corresponds to the view of that PE.

The GICR_DPRIR register is banked by view. Each PE must use the GICR_DPRIR that corresponds to the view of that PE.

SPI operation

In view 0, the view 0 control firmware must program the GICD_IVIEWRn registers so that it assigns each SPI to a view. SPIs can be routed only to PEs that are programmed to use the same view.

SPIs only matching affinities of other views are treated as SPI_NO_DEST_*. For 1 of N SPIs, the GIC only selects from PEs that are in the same view.

During GIC operation, if the view 0 control firmware wants to change an SPI assignment with GICD_IVIEWRn, then the interrupt must first be disabled using GICD_ICENABLER. To disable the SPI, we recommend that software coordinates with the original view of the interrupt, and poll its own GICD_CTLR.RWP, before allowing the view 0 control firmware to allocate the SPI to a different view. In view 0, GICR_CTLR.RWP does not guarantee that interrupts from other views have been successfully returned from their old processors. Therefore, if view 0 does not coordinate with the old view or directly use the GICD registers of the old view, then the view 0 control firmware must ensure that the interrupt can be successfully delivered to the new view, before considering the operation complete.

SPI error recovery

Multi view adds an extra error bit for each SPI, that is, the view error bit in the 5.2.29 GICD_ICVERRRn, Interrupt Clear View Error Registers on page 177.

The group error bit and error bit are set when software writes one to GICD_ISERRRn, in the view that corresponds with the SPI view assignment of the GICD_IVIEWRn register.

The view error bit in GICD_ICVERRRn is set when software in view 0 writes one to GICD_ISERRRn.

Error recovery in view 0 must clear the view error bit and then reprogram GICD_IVIEWRn for the SPI, before exiting error recovery. Deactivates from PEs in a different view are not applied.

LPI and virtual operation

LPI support and virtual support are present for view 1 only.

The GIC treats register accesses from view 2 and view 3 as if no LPI support is present.

If mapping or move commands occur to PEs that are in view 2 or view 3, then the commands fail.

SGI operation

SGI broadcasts are broadcast only to PEs that are in the same view as the PE that generates the SGI.

SGI generation fails if the destination is in a different view to the source.

PMU operation

The view 0 control firmware can program the GICD_SAC.GICPVIEW field, so that it assigns the GICP page to view 0 or one of the three functional views.

When the GICP page is assigned to view 0, the PMU records the events from all views. When a nonzero view is assigned, the PMU only records the events that correspond to that view.

Trace operation

The GIC records the RAS error record 0 errors from all views when GICT_ERR<n>CTLR.DIS_NZM == 0. If the view 0 control firmware sets DIS_NZM to 1, then the GIC records the RAS error record 0 errors from view 0 only.

The RAS error records for RAM errors are viewable by view 0 only.

4.6 RAMs and ECC

The GIC-720AE uses multiple RAMs to store a range of states for all types of interrupt. In typical operation, the RAMs are transparent to software.

Each RAM is protected from errors using an ECC with *Single Error Correction and Double Error Detection* (SECDED).

If single or double errors are detected, they are reported in the software visible error records, see 4.17 Reliability, Accessibility, and Serviceability on page 97 for more information.

4.6.1 RAM error simulation

For each RAM, software can use a GICx_ERRINSR register to simulate a transient ECC single-bit or double-bit error.

The GICR_ERRINSR applies to the RAM in the *GIC Cluster Interface* (GCI).

The GICD_ERRINSRn applies to the following RAMs:

0	SGI RAM
1	SPI RAM 0
2	SPI RAM 1
3	SPI TGT RAM
4	SPI LPI RAM
5	LPI RAM bank 0
6	LPI RAM bank 1
7	LPI RAM bank 2
8	LPI RAM bank 3
9	*Pending Table System* (PTS) RAM
10	*Virtual ITS Communication Module* (VICM) RAM
11	VSPA RAM
12	vTGT residency RAM
13	vTGT store RAM
14	vTGT search RAM
15	CC RAM

These registers cause an error to be inserted, to a specified address and location in the associated RAM. The ECC encoder and decoder are checked but the RAM content is not modified. These registers are all Secure access only, unless Secure software sets GICD_SAC.GICTNS to 1, to allow Non-secure access.

After software inserts an error, the GIC reports the error in the associated error record, in the same manner as a normal ECC error. However, the software injected error has no effect on the functionality of the GIC, so software can inject errors injection during operation.

If a co-incident real error occurs, then the GIC reports the real error instead and triggers the normal containment mechanism for that interrupt type.

Related information

GICD_ERRINSRn, Error Insertion Registers on page 169
GICR_ERRINSR, Error Insertion Register on page 218
GITS_C_ERRINSR, Error Insertion Collection cache register on page 244
GITS_D_ERRINSR, Error Insertion Device cache register on page 241
GITS_V_ERRINSR, Error Insertion Event cache register on page 242

4.6.2 Scrub

The GIC-720AE holds significant programming and interrupt states in RAM, which is protected by *Single Error Correction and Double Error Detection* (SECDED).

However, some RAM contents might be static for a long duration, and there is a potential for errors to accumulate if a particular address is not periodically accessed. To prevent this occurring, software can periodically trigger a low-priority scrub of a RAM, by setting the GITS_FCTLR.SIP, GICR_FCTLR.SIP, and GICD_FCTLR.SIP bits. This process triggers a check and if necessary, a write-back of all valid RAM entries. Any errors that are found during a scrub are also reported in the relevant RAS error record.

4.7 Direct injection

The GIC-720AE supports direct injection of SGIs (vSGIs) and LPIs (vLPIs) into virtual machines, without the processor needing to change state to execute the hypervisor in a virtualized system.

To support these features, the GIC must be configured with the following parameters:

- `gicv41_support` = 1
- `lpi_support` = 1.
  vSGI support requires `lpi_support` to be enabled because the GIC uses some ITS functionality to process vSGIs.

To map vPEs within the GIC, software must use the ITS `VMAPP` command.

The GIC-720AE requires the use of the Valid (V) and Allocate (A) bit in the VMAPP command. Behavior is unpredictable if any of the following occur:

- Use of VMAPP(V1A1) command when any mapping already exists for the vPE.
- Use of VMAPP(V1A0) command before a VSYNC has completed after a VMAPP(A1V1) command for the same vPE.
- Use of VMAPP(V0A0) command while mappings exist on any other ITS for the same vPE.
- The valid data fields of the all VMAPP commands for vPEs are not the same, excluding the RDbase.

For more information, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

To maintain information about the mapped vPEs, the GIC uses a single vPE table for each chip that is shared between all Redistributors and ITSs on the chip (GITS_TYPER.SVPET != 0). The GIC considers the table to be allocated when either:

- The first GICR_VPROPBASER.Valid bit is set.
- The first GITS_CTLR.Enabled bit is set on an ITS with GITS_BASER2.Valid == 1.

After the vPE table is allocated, the vPE table must be private to the GIC. The behavior is unpredictable if:

- The vPE table is modified while allocated.
- The vPE table is nonzero and not previously flushed out by this GIC.

In a multichip configuration, each chip must have a unique vPE table.

The properties for the chip-wide table are then taken from either the GICR_VPROPBASER or GITS_BASER2 register. Software can read the GICR_VCFGBASER register to discover properties of the chip-wide table.

The GIC does not relinquish control of the memory until all GICR_VPROPBASER registers, and all GITS_BASER2 registers on enabled ITSs, do not point at the vPE table. Any attempt to program mismatched values does not update the current programming and may be reported as an error.

4.7.1 Doorbells

Doorbell interrupts are physical LPIs that indicate to the hypervisor that an interrupt is available for that vPE.

Each vPE can be programmed with a unique doorbell using the ITS VMAPP or VMOVP command. When the first vLPI or vSGI becomes pending for a vPE, the GIC generates a single doorbell interrupt for that vPE. The doorbell interrupt is then masked until the vPE becomes resident.

The GIC-720AE has the following doorbell characteristics:

- Doorbell IDs must be unique and not mapped to any DeviceID and EventID on any ITS.
- GIC-720AE does not support individual doorbells, so GITS_TYPER.nID == 1.

- Doorbells only generate if the relevant virtual group enable is set when the vPE was last made resident. The vPE has not been made resident since being mapped, see 4.7.2 Residency and VMOVP on page 68.
- The GIC ignores and reports `VMAPP` and `VMOVP` commands that specify a doorbell ID that is outside of the range of GICR_PROPBASER.IDbits.

Doorbell properties are cached when mappings are first made. You can change the properties by updating the LPI Configuration table and issuing a `VINVDBL` command from any ITS that has the vPE mapped.

Doorbell properties are not transferred between chips and are refetched if a `VMOVP` moves a vPE to a remote chip.

Software must not disable GICR_CTLR.EnableLPIs bits while the vPE mapping exists, because this prevents doorbells fetching properties, and they are cached as disabled.

4.7.2 Residency and VMOVP

Software freely moves vPEs around between PEs on both the local and remote chips, using the ITS `VMOVP` command.

The GIC-720AE only supports GITS_TYPER.VMOVP ==1. Therefore, the `VMOVP` command must run only on a single ITS (with a mapping to the vPE) for the GIC to ensure that all ITS in the system seamlessly continue to deliver interrupts to the vPE. The GIC only updates each ITS when it next accesses the vPE.

Doorbells are delivered to the PE that the vPE is mapped to. For more information on doorbells, see 4.7.1 Doorbells on page 67. However, the GIC supports making vPEs resident (to deliver interrupts) on any PE on the same chip, as the current mapping without a `VMOVP`. That is, there is no need to issue `VMOVP` before making a vPE resident, unless either moving chip or balancing the doorbells across different PEs.

To deliver the interrupt to a vPE, it must be made resident on a PE, which is completed by using GICR_VPROPBASER of the relevant PE and polling for GICR_VPROPBASER.Dirty == 0. The GIC attempts to deliver any interrupt to the PE before dropping the GICR_VPROPBASER.Dirty bit.

We recommend that the CPU Virtual Group enabled are set before making a vPE resident. This is to ensure the GIC can enforce that an interrupt has reached the PE and therefore prevents a race with the GuestOS reaching WFI before the interrupt is delivered. The behavior is unpredictable if software attempts to do any of the following:

- Make a vPE resident on multiple PEs.
- Map a vPE resident when not mapped to PE on the same chip.
- Issue a `VMOVP` command to a resident vPE.

The GIC is designed to ensure that the highest priority interrupt is always ready, waiting for when a vPE is made resident. However, there are two bits in GICD_FCTLR2 that control whether the residency change is delayed under certain conditions:

GICD_FCTLR2.RWC – Residency wait during command

If an LPI command is active, which could make an interrupt available for a vPE, the GIC does not stall the residency handshake, unless RWC is set.

GICD_FCTLR2.RWS – Residency wait during search

Under heavy load, LPIs are sent to the PT. Under extremely heavy load, and when a vPE has been recently resident or when LPI commands run, it is possible that the highest priority interrupt for a vPE has not yet been retrieved. To ensure fast residency changes, the GIC does not wait on PT searches for residency, unless RWS is set.
Sometimes, specifically for *Double Error Detection* (DED) errors or `INVALL` commands, setting GICD_FCTLR2.RWS leads to a significant increase in the latency of the residency handshake.

Interrupts found under the two previous conditions are delivered when they are found and are the highest priority, if the vPE is still resident. If the vPE is taken out of residency, a new doorbell is generated, if enabled.

We recommend that the RWC and RWS bits are not set during normal operation.

When making an interrupt non-resident by writing GICR_VPROPBASER.Valid == 0, then GICR_VPROPBASER.PendingLast indicates whether there are remaining interrupts for the vPE. If it is set after writing GICR_VPROPBASER.Dirty == 0, then the doorbell remains masked and software must make the vPE resident again at some point in the future.

If GICR_VPROPBASER.PendingLast is written to 1 when GICR_VPROPBASER.Valid is cleared, the GIC optimizes the residency handshake and leaves the doorbell masked without checking if there are more interrupts for the vPE.

4.7.3 Errors and debug

The vPE Configuration table is stored in RAM and backed up in the main memory. If corruption occurs during accesses to the vPE table (or virtual Pending tables), then the error is recorded in error record 0, with one of the SYN_VPE_CFG* error syndromes.

If this corruption occurs, or error record 0 overflows, use the GICR_VERRR register to check the status of vPEs by completing a `FIND` command.

If any vPE is marked as errored, then it has become corrupted and software must flush out the error.

If not resetting the GIC, the vPE can be flushed out of the GIC by doing the following on the GICD where GICR_VERRR was read:

1. Issue `VMAP(V0A0)` commands on all ITSs.
2. Issue `VMAPP(V1A1)` on one ITS with a vPT_size of at least as large as the original.
3. Issue `VMAPP(V0A1)` on the same ITS to flush out everything.
4. Clear the error using the GICR_VERRR `CLR` command.
5. Repeat the GICR_VERRR `FIND` command until it indicates no errors.

6. Recreate vPE as normal with a new vPT.

GICR_VERRR can also be used to set errors for software test purposes, and to read a range of data stored in the GIC about a vPE.

4.8 Low latency support

GIC-720AE can be integrated into systems that require interrupts to be distributed to real-time peripherals with a deterministic low latency. To support this requirement, GIC-720AE provides up to 960 real-time SPIs and up to 48 real-time PPIs.

The GIC-720AE uses an inbuilt dedicated fast path to deliver the *Highest Priority Pending Interrupt* (HPPI) to a target core. The GIC assigns an interrupt as the HPPI when all the following conditions apply:

- The interrupt is pending, enabled, and not active.
- The interrupt is routed to a PE that connects to a real-time GCI on the same GICD.
- The corresponding group enables are enabled in the GIC and processor.
- The interrupt is the highest priority interrupt among all other valid interrupts to a particular core.


Note

- For SPIs with the same priority, the GIC considers the lower ID as higher in priority. Therefore, we recommend spreading out latency critical interrupt with the same priority across several cores, otherwise higher ID interrupts are not delivered until the lower ID interrupt is delivered and activated.
- For PPIs and SGIs, the GIC makes a random selection between available interrupts with the highest priority.
- Real-time interrupts can route to other PEs, including PEs on remote chips, but the latency guarantees are not maintained.

To achieve low latency for interrupts with the Cortex®-R82AE processor, we recommend that GICR_FCTLR.ECP remains set to 1.

The GIC does not provide interrupts with a deterministic low latency in the following situations:

- An interrupt that is not a valid HPPI. These interrupts are not sent to the core until they become HPPI.
- Interrupts that are configured as 1 of N interrupts.
- Message interrupts that GICD_SETSPI_SR, GICD_SETSPI_NSR, GICM_SETSPI_SR, or GICM_SETSPI_NSR generate.
- Interrupts that target a powered down core or a core that is in the process of powering down.
- An interrupt whose attributes change, including pending value, while the interrupt is HPPI.
- Interrupts that are generated when the GIC is in Q-Channel low-power state.

- When the core exerts back pressure on the GIC Stream interface, between the GIC and a core, because the core is busy.

The latency value of an interrupt through the GIC also depends on the processor response delay and any backpressure that the core exerts on the GIC Stream interface. To get low interrupt latency, it is important that the processor does not delay its response when two sets are outstanding, or when a clear is outstanding on the GIC Stream interface.

4.9 SGIs

Software Generated Interrupts (SGIs) are inter-processor interrupts, that is, interrupts generated from one core and sent to other cores.

Each core, or vPE if configured, in the system processes an SGI independently of the other cores. The priority of an SGI, and other settings, are also independent for each core.

Physical SGIs are generated by writing to System registers in the CPU interface of the core that generates the interrupt. SGIs are edge triggered.

Up to 16 SGIs can be recorded for each target core or vPE, where each SGI has a different INTID in the ID0-ID15 range.

4.9.1 SGI programming

The generation of an SGI depends on whether the SGI is physical or virtual.

Physical SGIs

To program a physical SGI, each processor can use its GICR register map. See 5.5 Redistributor registers for SGIs and PPIs summary on page 207.

Virtual SGIs

To program a virtual SGI, software can issue a vSGI ITS command.

Software can also program the vSGIs by writing to the virtual Pending table of a vPE, and then issuing a `VMAPP` command to allocate the memory to the GIC. After issuing `VMAPP` command, software must not write to the virtual Pending table to attempt to generate a virtual SGI. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.

4.9.2 SGI direct injection

Software can directly inject SGIs by writing the vPE and SGI-INTID to the GITS_SGIR register.

The GITS_SGIR register is always accessed using the GICD ACE5-Lite subordinate interface, even in a distributed system.

Unlike their physical equivalent, vSGIs do not have an active state, so no deactivation is required.

If the vPE is not mapped on the ITS that the GIC uses to generate the vSGI, then the GIC generates a debug error. See 4.17.4.3 SGI RAM error records 3-4 on page 106.

4.9.3 SGI multichip

When chips are connected, then physical SGIs are routed to the destination chip based on the target affinity. Virtual SGIs are routed to the target vPE, irrespective of the chip that the vPE is currently mapped to.

4.9.4 SGI error recovery procedure

If an uncorrectable SGI error occurs, then software must clear the error for that interrupt. After clearing the error, software can reprogram the interrupt to the intended settings.

For uncorrectable errors that occur in the SGI RAM, software is required to perform the following recovery sequence:

1. Read the error record, to determine if an uncorrectable error has occurred.
2. Clear the error record, to enable future errors to be tracked.
3. Read all GICR_ICDERRR registers, so that you can identify the SGIs that have errors. The GICR_ICDERRR registers must be read from the Secure state.
4. If necessary, read out any of the current programmed states. This includes programmed data that is corrupted and generates an error, unless GICT_ERR0CTLR.UE is disabled. We recommend that the intended programming is stored in memory, so that this step is not required.
   The GICR_NSACR is overwritten when an error occurs, so the pre-error value cannot be read back at this stage.
5. Write to GICR_ICENABLER0, to disable all interrupts that have errors.
6. Write 1 to the GICR_ICDERRR bits that step 3 on page 72 indicates are showing an SGI error. This write clears the interrupt error and reverts the corresponding GICR_IGROUPR0, GICR_IGRPMODR0, and GICR_NSACR bits to their default values as programmed in the corresponding bits of GICR_SGIDR.
7. Reprogram the interrupt to the intended settings.
8. Re-enable the reprogrammed interrupts by writing to the relevant GICR_ISENABLER0.
9. Recheck the error record, to ensure that no more errors are reported. If necessary, repeat the recovery sequence from step 2 on page 72.

While errored, the GIC uses the values in GICR_SGIDR to determine if SGIs are generated.

The GIC does not provide a GICR_ISDERRR register, so you cannot set errors on the SGI RAM.

Related information

SGI RAM error records 3-4 on page 106

4.10 PPIs

A *Private Peripheral Interrupt* (PPI) identifies an interrupt source, such as a timer, that is private to the core, and which is independent of the same source for another core. PPIs are typically used for peripherals that are tightly coupled to a particular core.

Interrupts that connect to the PPI inputs associated with one core, are sent only to that core. Each core processes a PPI independently of other cores. The settings of a PPI are also independent for each core.

A PPI is unique to one core. However, the PPIs to other cores can have the same INTID. Up to 48 PPIs can be recorded for each target core, where each PPI has a different INTID in the ID16-ID31 or ID1056-ID1087 range. To use the ID1056-ID1087 range, the core must support the GICv3.1 extensions.

PPI signals are active-LOW level-sensitive by default. However, you can set a PPI signal to be either level-sensitive or edge-triggered using GICR_ICFGR1, GICR_ICFGR2E, and GICR_ICFGR3E. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.

The GIC-720AE provides an option, through parameters, to include a synchronizer or inverter, or both, on each PPI interrupt signal. See 3.2.4 GCI PPI signals on page 40 for more information.

For information about the purpose of each PPI used by the processor core in your system, refer to the processor Technical Reference Manual.

4.10.1 PPI signals

Each PPI is a physical interrupt signal that can be configured to be either a level-sensitive interrupt or an edge-triggered interrupt.

The two configurations of physical PPI signal are:

Level-sensitive

The interrupt is pending while the interrupt input is asserted. As with previous Arm GICs, PPIs are active-LOW by default. However, you can change these default settings, see 4.1 Interrupt types on page 58 for more information.

Edge-triggered

A rising-edge on the interrupt input causes the interrupt to become pending. The pending bit is cleared later when the interrupt is activated by the CPU interface.

To set the correct settings for the system, you must program the GICR_ICFGR1, GICR_ICFGR2E, and GICR_ICFGR3E registers.

For more information, see the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4* and the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

4.10.2 PPI programming

To program a physical PPI, each processor can use its GICR register map.

Related information

Redistributor registers for SGIs and PPIs summary on page 207

4.10.3 PPI direct injection

The GIC-720AE cannot directly inject PPIs into vPEs.

4.10.4 PPI multichip

Multichip operation does not affect PPIs.

4.10.5 PPI error recovery procedure

If an uncorrectable PPI error occurs, then software must clear the error for that PPI. After clearing the error, software can reprogram the interrupt to the intended settings.

Note

This recovery procedure is also applicable to SGI programming in the *GIC Cluster Interface* (GCI) RAM.

PPI priority values can be stored in a RAM inside the GCI and is protected with ECC.

If an uncorrectable error occurs, the GIC uses the default priority values that GICR_DPRIR specifies for the relevant interrupt group, and it continues to deliver the interrupt as normal but with a default priority.

Errors affecting PPIs are reported in GICT error records 7-8. See 4.17.4.5 PPI RAM error records 7-8 on page 108.

Software can determine the errored PPI IDs by reading GICR_ISERR0 or GICR_ISERR1E.

Software can clear PPI errors by rewriting the relevant priority field in GICR_IPRIORITYRn or GICR_IPRIORITYnE, or by writing to GICR_ICERR0 or GICR_ICERR1E, in which case the priority takes the relevant default value that GICR_DPRIR specifies.

If a GICD_IPRIORITYR register of a corrupted PPI is read, then the corrupted data is made available. This data is reported in error record 0 with a SYN_GICD_CORRUPTED error syndrome. If GICT_ERR0CTLR.UE == 1, then the GIC issues an SLVERR ACE5-Lite bus error.

For debug purposes, software can trigger these error cases by writing to GICR_ISERR0 or GICR_ISERR1E. To test the ECC error reporting, software can use GICR_ERRINSR.

Related information

PPI RAM error records 7-8 on page 108

4.11 SPIs

A *Shared Peripheral Interrupt* (SPI) is generated by a peripheral that is accessible across the whole system such as a USB receiver, and which can connect to several cores. The GIC supports real-time SPIs. SPIs are typically used for peripherals that are not tightly coupled to a specific core.

You can program each SPI to target either a particular core or any core. Activating an SPI on one core activates the SPI for all cores. That is, the GIC-720AE allows at most one core to activate an SPI (cannot be activated by multiple cores). The settings for each SPI are also shared between all cores.

SPIs can be either:

Real-time SPI

Real-time SPIs are generated by wire inputs. The GIC-720AE can support up to 960 real-time SPIs, which connect to the rlt_spi input signals on the Distributor. The number of real-time SPIs depends on the implemented configuration. The real-time SPIs are the lowest SPI IDs that are assigned to the chip. For a single-chip configuration, the first SPI has an ID number of 32, and the permitted ID values are in steps of 32, from ID32 to ID991.
You can configure whether each real-time SPI is triggered on a rising edge or is active-HIGH level-sensitive. The GIC-720AE provides a build-time option, to include one or both of a synchronizer or inverter for each rlt_spi signal.

Standard SPI

The GIC-720AE can support up to 1984 standard SPIs but this quantity decrements for each real-time SPI that is present. During configuration of the GIC, you can allocate some or all standard SPIs to be message-based or you can set all standard SPIs to be a physical spi signal:

Message-based

Software can generate these SPIs by writing to the ACE5-Lite subordinate programming interface. The INTIDs for message-based SPIs commence after the last INTID of the physical spi range of INTIDs.

Physical wires

A physical wire interrupt connects to an spi signal on an SPI Collator, and the number of SPI Collators is configurable. Each SPI Collator has a limit of 1024 spi signals. The first ID for a standard SPI is, INTID = 32 + (number of real-time SPIs). Depending on the configuration, the possible INTIDs are ID32-ID991 and ID4096-ID5119.
You can configure whether each standard SPI is triggered on a rising edge or is active-HIGH level-sensitive. The GIC-720AE provides a build-time option, to include one or both of a synchronizer or inverter for each spi signal.

If an SPI ID is allocated as a physical spi input signal, then software can still use that SPI ID as a message-based SPI, provided that the hardware ensures that the spi signal is held to a logic level that represents the inactive state.

The SPI Collator converts wire-based interrupts into messages to reduce system wiring, and to allow more aggressive clock gating of the GIC to reduce power consumption. See 3.5 SPI Collator on page 50 for more information.

SPIs are programmed through the GICD register address space, which is spread coherently across all configured chips to provide a single view to the *Operating System* (OS).

You can trigger a valid SPI by using the GICD_SETSPI_NSR or GICD_SETSPI_SR registers, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

4.11.1 SPI signals

Each SPI is a physical interrupt signal that can be configured to be either a level-sensitive interrupt or an edge-triggered interrupt. The standard spi signals connect to an SPI Collator and the real-time rlt_spi signals connect to the Distributor.

The two configurations of physical SPI signal are:

Level-sensitive

The interrupt is pending while the interrupt input is asserted. As with previous Arm GICs, SPIs are active-HIGH by default. However, you can change these default settings, see 3.5.2 SPI Collator wires on page 50 and for real-time SPIs see rlt_spi signal inversion and synchronization on page 76 for more information.

Edge-triggered

A rising-edge on the interrupt input causes the interrupt to become pending. The pending bit is cleared later when the interrupt is activated by the CPU interface.

To set the correct settings for the system, you must program the GICD_ICFGRn or GICD_ICFGRnE registers. For more information, see the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4* and the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

The GIC-720AE provides optional synchronizers on every interrupt wire input. The GIC also provides return signals, spi_r and rlt_spi_r, to support interrupt pulse extension when sending edge-triggered interrupts across domain boundaries, see 3.5.2 SPI Collator wires on page 50 and for real-time SPIs see rlt_spi signal inversion and synchronization on page 76.

rlt_spi signal inversion and synchronization

The real-time SPI inputs can be inverted and synchronized to the GICD clk signal when the appropriate build-time options are set. The GICD also provides SPI outputs that can be used to create interrupt pulse extension for edge-triggered interrupts that cross clock domains.

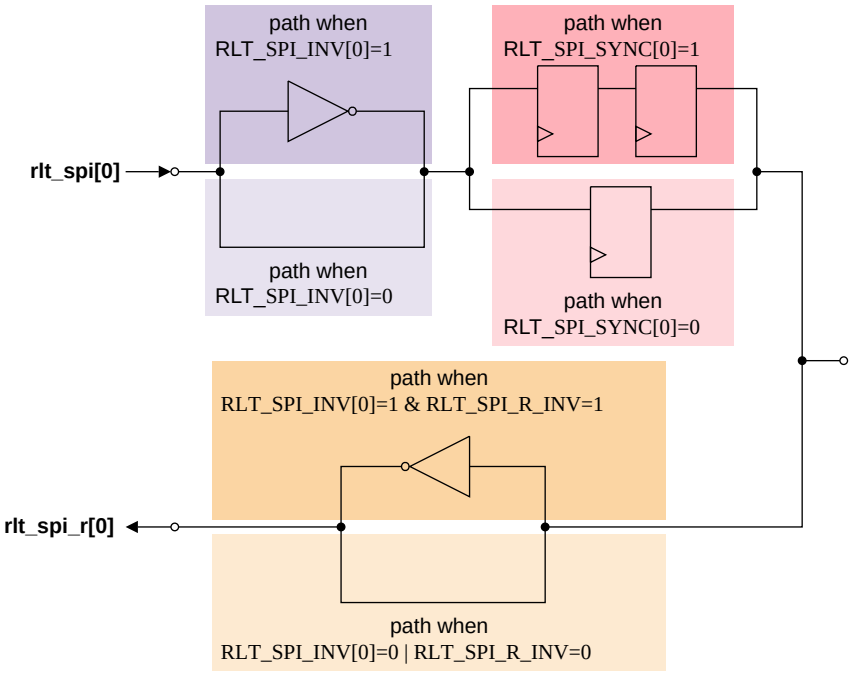
By default, the asserted level of a real-time SPI is active-HIGH. However, each real-time SPI can be either inverted, synchronized, or both, using the `RLT_SPI_INV[n]` and `RLT_SPI_SYNC[n]` build-time options, where:

- `RLT_SPI_INV[n]` == 1 indicates that the inverter is enabled.
- `RLT_SPI_SYNC[n]` == 1 indicates that the synchronizer is enabled.
- `[n]` = SPI_ID – 32.

Each real-time SPI input signal, rlt_spi, has a corresponding rlt_spi_r signal after the synchronizer or capture flop that can be used to create pulse extenders for edge-triggered interrupts that cross clock domains. If `RLT_SPI_INV[n]` is set to 1, then the signal after the synchronizer is inverted with respect to the input unless the `RLT_SPI_R_INV` option is set to 1. If the `RLT_SPI_R_INV` option is set to 1, then it removes any inversion that `RLT_SPI_INV[n]` applies to individual SPIs.

The following figure shows the effect of the `RLT_SPI_INV[0]`, `RLT_SPI_SYNC[0]`, and `RLT_SPI_R_INV` build-time options on the rlt_spi[0] signal.

Figure 4-2: Real-time SPI build-time options and signal conditioning



4.11.2 SPI programming

To program an SPI, each processor can use the GICD or GICDA register map.

Related information

Distributor registers (GICD/GICDA) summary on page 135

4.11.3 SPI routing and 1 of N selection

If GICD_TYPER.No1N==0, then the GIC-720AE supports 1 of N selection of SPI interrupts. You can program an SPI to target several cores, and the GIC-720AE can select which cores receive an SPI.

When the relevant GICD_IROUTERn.Interrupt_Routing_Mode == 1, the GIC selects an appropriate core for an SPI.

When GICD_IROUTERn.Interrupt_Routing_Mode == 0, the SPI is routed to the core specified by the remaining fields of GICD_IROUTERn.

The GIC-720AE only sends an SPI to cores that are powered up and have the relevant interrupt group enabled. The GIC-720AE prioritizes cores that are considered active, but if there are no active cores, it selects inactive cores.

The selections that the GIC-720AE makes can be controlled or influenced by several 1 of N features:

cpu_active signal

A cpu_active signal is an input to a Redistributor that corresponds to a particular core. When a cpu_active signal is LOW, it indicates to the GIC that a core is in a transparent low-power state such as retention, and that it must be selected as a target for an SPI if there are no other options possible.
Ideally, the cores that are in retention are not woken without explicit software intervention, so that cores spend more time in retention. To ensure that this behavior is usually the case, use the following guidelines:

- Cores in retention must drive their corresponding cpu_active signal LOW.
- Powered-up cores that are not in retention must drive their cpu_active signal HIGH.

Typically, a power controller or power control logic generates the cpu_active signal. If this signal is not available in the system, the input must be tied HIGH.


Note

- When a core is powered down, the value of its cpu_active signal is irrelevant. This irrelevancy is because the software programming requirements for the GIC ensure that it knows when cores are powered up or down.
- The cpu_active signal provides an indication only, it cannot stop selection of the core or stop the GIC sending messages to the core.

GICR_CTLR.DPGxx (Disabled Processor Group)

Setting a DPG bit prevents 1 of N interrupts of a particular group being sent to that core. Any interrupts that have not reached a core at the time of the change, are recalled and reprioritized by the GIC.

Processor and GICD group enables and GICR_WAKER.ProcessorSleep

A 1 of N interrupt is not sent to a core if one of the following is true:

- The core is asleep, as indicated by GICR_WAKER.ProcessorSleep.
- The interrupt group is disabled by either the processor or the GICD_CTLR group enables.

Interrupt class

This is an implementation-defined feature that the GIC-720AE provides. Each core can be assigned to either class 0 or class 1 by writing to the relevant GICR_CLASSR register. An SPI, programmed as 1 of N, by GICD_IROUTERn.Interrupt_Routing_Mode, can be programmed to target either class 0, class 1, or both classes by the GICD_ICLARn register. By default, all 1 of N SPIs can go to both classes, so the interrupt class feature is disabled by default. The system can use this partitioning for any purpose, for example in an Arm® big.LITTLE™ system, all the big cores can be in class 1 and little cores in class 0, allowing 1 of N SPIs to be partitioned according to the amount of processing they require.

GICD_CTLR.E1NWF

The GICD_CTLR.E1NWF bit controls whether the GIC-720AE wakes a core if there are no other possible targets for a 1 of N SPI.
The GIC tries to wake the minimum of cores possible and only wakes a core if there is no other possible target awake that is able to accept the 1 of N interrupt. Therefore, the GIC uses the GICR_CTLR.DPG and GICR_CLASSR.Class bits to determine if any core is awake that can accept the interrupt. If a suitable core is not awake, the GIC then wakes a core.

We strongly recommend that if you use GICD_CTLR.E1NWF, you must also set the GICR_CTLR.DPGx bits to specify whether a core is likely to accept a particular interrupt group in a timely manner. The GIC does not continue to wake cores until one is found. The GIC-720AE uses two passes to try to find the best place for a 1 of N interrupt, by using a round-robin arbiter between:

- Any core that has its cpu_active signal set, is fully enabled for the interrupt, and has no other pending interrupts.
- Any core that is fully enabled for the interrupt and has no interrupts of a higher priority than the 1 of N interrupt.

If neither option is available to the 1 of N, the interrupt is assigned to any legal target and regularly re-evaluated to ensure that it is not excluded from other SPIs of the same priority.

4.11.4 SPI direct injection

The GIC-720AE cannot directly inject SPIs into vPEs.

4.11.5 SPI ownership for multichip operation

The owner of an SPI block is defined by the GICD_CHIPR<n> registers.

You can remove SPI blocks from a chip and add them to another chip by reprogramming the relevant GICD_CHIPR<n> registers during operation. As with all Routing table operations, GICD_DCHIPR.PUP must be polled to check completion of the operation.

Before you change the owner of an SPI block, you must ensure that the GICD_CTLR group enables have cleared, GICD_CTLR.RWP has returned to 0, and that the SPI blocks are removed from a chip before they are added to another chip.

When an SPI block is removed from, or added to, a chip, all programming that is associated with the SPI block returns to the reset state.

You must not alter the SPI_BLOCK_MIN of an online chip because the results are unpredictable. To change SPI_BLOCK_MIN:

1. Move the chip offline by setting GICD_CHIPR<n>.SocketState = 0.
2. Alter SPI_BLOCK_MIN when the chip is brought back online.

4.11.6 SPI operation for multichip operation

When the Routing table is set up, SPIs can be programmed through any connected chip, and accesses to update stored values are routed over the cross-chip interface of the chip that owns the SPIs.

SPIs can be routed to remote chips by programming the relevant GICD_IROUTERn register. Remote chips are targeted using either Affinity2 or Affinity3, and the affinity level can be discovered using GICD_CFGID.AFSL.

If SPIs within an SPI block are sent to multiple chips, we recommend that you do not read or write the GICD_ISACTIVERn(E), GICD_ICACTIVERn(E), GICD_ISPENDRn(E), and GICD_ICPENDRn(E) registers. It is inefficient and these registers are not needed for immediate operation.

You can set interrupts to pending by writing to GICD_SETSPI_NSR, GICD_CLRSPI_NSR, GICD_SETSPI_SR, and GICD_CLRSPI_SR. For efficient operation, we recommend that sources are programmed to write SPI IDs that their chip owns. Other SPI IDs are supported if these SPIs are owned somewhere in your system.

By default, the GIC-720AE does not guarantee that the pending bit has reached the point of serialization for writes to set interrupts pending. This behavior means that there is a race between the pending bit being set and an activate being processed by the GIC after the bresp signal asserts. To ensure that writes always propagate to the point of serialization, set GICD_FCTLR.POS = 1.

SPI Collators in multichip

The SPI Collator wires are always connected to the lowest owned SPIs on the chip.

For example, if GICD_CHIPRn.SPI_BLOCK_MIN = 4, the SPI Collator wires to chip-`n` drive SPI IDs that start from 160, which is calculated by (4 × 32) + 32 = 160. Therefore, in a homogeneous 2-chip system, each chip must not use more wires than 16 × (the number of configured SPI blocks).

SPI 1 of N

The GIC-720AE never sends a 1 of N SPI to another chip.

4.11.7 SPI error recovery procedure

If an uncorrectable SPI error occurs, then software must clear the error for that SPI. After clearing the error, software can reprogram the interrupt to the intended settings.

If an SPI has an uncorrectable error, GICD_ICERRRn identifies the SPI. While in this error state, the interrupt reverts to a disabled, Secure Group 0, edge-triggered SPI, which is already pending.

For uncorrectable errors, software is required to perform the following recovery sequence:

1. Read the error record, to determine if an uncorrectable error has occurred.
2. Clear the error record, to enable future errors to be tracked.
3. Read all GICD_ICERRRn registers, so that you can identify the SPIs that have errors. The GICD_ICERRRn registers must be read from the Secure state. See step 6 on page 81
   If the error record reports only one error, the block that contains the error can be determined using the ID in the GICT_ERR2MISC0 register, by calculating the block number as 1 + (ID / 32). However, if an overflow occurs, software must check all GICD_ICERRRn registers.
4. If necessary, read out any of the current programmed states. This includes programmed data that is corrupted and generates an error, unless GICT_ERR0CTLR.UE is disabled. We recommend that intended programming is stored in memory so that this step is not required.
5. Write to GICD_ICENABLERn, to disable all interrupts that have errors.
6. Write 1 to the GICD_ICERRRn bits that step 3 on page 81 indicates are showing an SPI error. This write clears the interrupt error and reverts the corresponding GICD_IGROUPRn, GICD_IGRPMODRn, GICD_ICFGRn, and GICD_NSACRn bits to their default values.
   If Secure software allows Non-secure software to clear an error for a Non-secure interrupt, it can first clear the error on the Secure data (GICD_GROUPn, GICD_GRPMODn, and GICD_NASCRn). The software uses the corresponding bit of the GICD_ICGERRn and must reprogram the three registers mentioned previously. Non-secure software is then allowed to read and clear GICD_ICERRR for those specific interrupts.
7. Read GICD_ICERRRn, to check that the error has cleared. If the error remains, then clear all the GICD_CTLR group enables so that it forces all SPIs to return to their owner chips. When GICD_CTLR.RWP returns to 0, repeat the write to GICD_ICERRRn. When the error clear is accepted, you can re-enable the group enables.
8. Reprogram the interrupt to the intended settings.
9. If the interrupt is reprogrammed to be level-sensitive, write to GICD_ICPENDRn to ensure that any edge-sensitive pending bits are cleared.
10. If the interrupt is edge-triggered, we recommend that software checks the device, if possible, in case an edge is lost.
11. Ensure that the active bit is set correctly depending on whether it is being processed. Clear the active bit using GICD_ICACTIVE to ensure that the interrupt is delivered when it is set to pending in the future. However, if the interrupt is being processed in a core, the interrupt might be delivered again before it is deactivated.
12. Re-enable the reprogrammed interrupts by writing to GICD_ISENABLER.
13. Recheck the error record, to ensure that no more errors are reported. If necessary, repeat step 2 on page 81.

To aid software debug, Secure software can use the GICD_ISERRn and GICD_ISERRRnE registers to insert error cases.

Related information

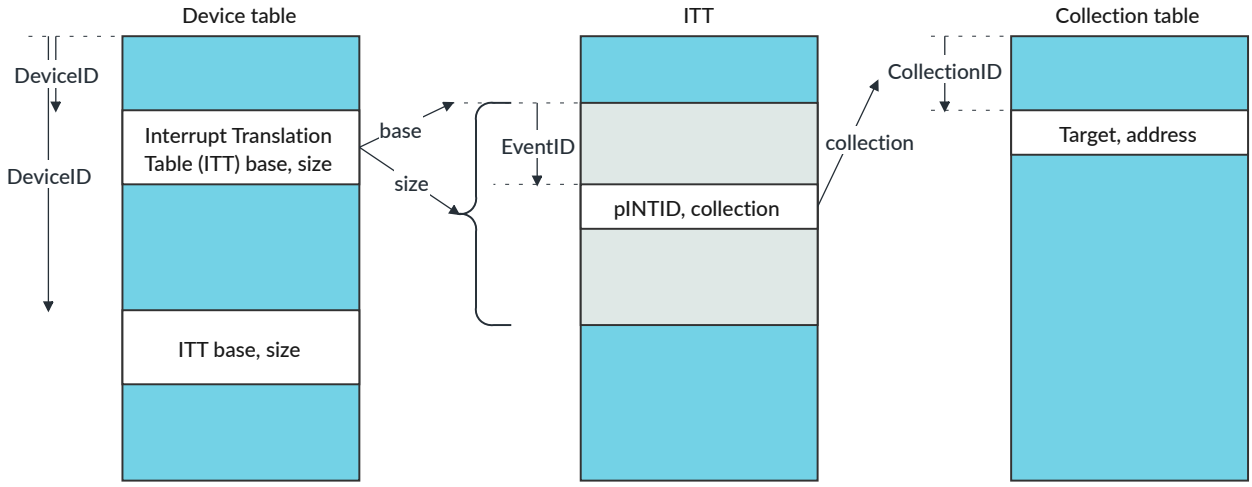
SPI RAM error records 1-2 on page 105

4.12 ITS

The GIC-720AE supports up to 32 *Interrupt Translation Services* (ITSs) for each chip. Each ITS is responsible for translating message-based interrupts from peripherals into LPIs or vLPIs.

Each ITS is compliant with the GICv3 and GICv4.1 architecture and is responsible for mapping translation requests with an EventID and DeviceID through to an INTID and target. The following figure shows the ITS process for a *physical INTID* (pINTID).

Figure 4-3: Physical ITS process



To reduce memory traffic and keep interrupt latency to a minimum, GIC-720AE has three 2-way set associative caches in each ITS:

DeviceID cache DeviceID to ITT base address
EventID cache DeviceID and EventID to collection
Collection cache Collection to target core

It is common for the DeviceID to be a non-contiguous number that is derived from the PCIe RequestorID. To ensure that this does not result in a sparse DeviceID table and wasted memory, the GIC-720AE supports indirect Device tables (GITS_BASERn.Indirect = 1) where the first-level table points at subtables that can be allocated at runtime. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.

The GIC-720AE uses memory-backed collections only, which means that before the ITS is enabled by writing to GITS_CTLR.Enabled, memory must be allocated for the Device table, the Collection

table, and the ITS command queue. To comply with the architecture, software must pre-clear these tables to 0, apart from pointers to cleared level-two Device tables, unless the tables were previously populated by the GIC-720AE.

When software uses GICv4.1 commands, it must provide a pointer to the chip-wide vPE table before enabling the ITS.

The GIC-720AE ITS supports all GICv3 and GICv4.1 commands that the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* describes.

GITS_TYPER.PTA is 0 for all configurations, which means that all references to processor cores in ITS commands are implemented through the GICR_TYPER.ProcessorNumber field.

Command and translation errors are reported through the RAS registers. See 4.17 Reliability, Accessibility, and Serviceability on page 97.

For information about how to program and use the ITS, see the *Locality-Specific Peripheral Interrupts, Arm® Generic Interrupt Controller v3 and v4*.

4.12.1 ITS cache control, locking, and test

The GIC-720AE can lock certain interrupt translations in the EventID cache.

If a translation is missed in a cache, several memory reads can be required to obtain the data necessary from memory. This behavior can result in a range of latency that might not be acceptable for some LPIs.

The GIC-720AE can lock certain translations into the ITS cache, with the following guarantee:

- Interrupts that are locked in ITS caches, always hit and never require any translation.

The ITS caches are automatically managed and invalidated as necessary when the GITS_BASERn registers are updated. Therefore, software intervention is not required. However, to aid debug and integration testing, you can force invalidation of the appropriate cache by setting the relevant bit in the GITS_FCTLR register.

A forced invalidation of the Event cache abandons all locked entries.

The GITS_OPR and GITS_OPSR registers control cache locking, when software provides the DEVICE_ID, EVENT_ID, and the correct GITS_OPR.LOCK_TYPE (ITS lock = 2). The GIC attempts to perform the lock, and reports the status in GITS_OPSR. If the lock succeeds, GITS_OPSR.REQUEST_COMPLETE == 1 and GITS_OPSR.REQUEST_PASS == 1.

Each cache set is 2-way set associative. Only one entry can be locked in each cache set. Any attempt to lock both ways in a set, reports as failed in GITS_OPSR. You can also use the GITS_OPR register to unlock entries that are locked.

The GITS_OPR register has two test features:

Trial Tests the mapping by writing a DeviceID and EventID to GITS_OPR with GITS_OPR.LOCK_TYPE = 1 (Trial). This causes the ITS to translate the supplied DeviceID and, or EventID pair, and report the generated translation data in GITS_OPSR. The GIC also reports whether the translation fails, GITS_OPSR.REQUEST_PASS == 0, or if it hit a locked entry, GITS_OPSR.ENTRY_LOCKED. The interrupt is not set to pending.

Track Can be used to detect the arrival of a certain EventID and, or DeviceID pair, which the GIC reports by setting GITS_OPSR.REQUEST_COMPLETE.

While any GITS_OPR operation, other than Track, is in progress, the GITS_OPSR.REQUEST_IN_PROGRESS bit is set and no further updates are accepted by GITS_OPR until the previous operation completes. To ensure that the operation is accepted, we recommend that the GITS_OPR value is read after writing. You can abort Track operation by writing GITS_OPR.LOCK_TYPE == Track abort.

4.12.2 MSI-64

The MSI-64 Encapsulator can be used to combine the DeviceID into single memory access writes to the GITS_TRANSLATER register in the ITS.

The ITS translates DeviceID/EventID pairs into LPI physical INTIDs.

A normal MSI/MSI64 write contains the EventID in the lower 16 bits or 32 bits of data. However, the DeviceID must be transported using a different method. The DeviceID is often derived directly from a PCIe RequestorID or *System Memory Management Unit* (SMMU) StreamID. If the EventID is greater than 16 bits, then 16-bit MSI writes are padded with zeros.

The GIC-720AE ITS supports two mechanisms:

awuser_*_s signal

If the `AWDEVICEID_FROM_AWUSER` build-time option is set to 1, the GIC takes the DeviceID from the awuser_*_s signal. You must ensure that rogue software cannot directly or indirectly, perform an access to the GITS_TRANSLATER register with a DeviceID that matches a real device.

MSI-64

When configured to support MSI-64, the ITS expects the DeviceID to be in the upper 32 bits of a 64-bit write to the GITS_TRANSLATER register.
To prevent rogue software accessing the GITS_TRANSLATER register and spoofing any device, we recommend that the GITS_TRANSLATER register is moved to an arbitrary page that is protected by the hypervisor.

The GIC-720AE has the following features to support this:

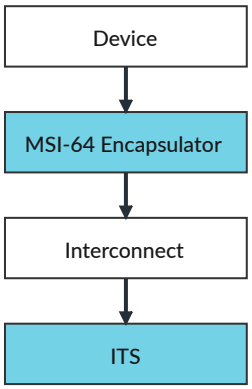
- The MSI-64 Encapsulator modifies the page address of accesses to the architectural GITS_TRANSLATER address, set by the msi_translator_page tie-off signal, to the system-defined page set by the msi64_translator_page signal.

- When the ITS shares an ACE5-Lite subordinate port, the its_transr_page_offset tie-off signal allows the GITS_TRANSLATER register page to be moved to anywhere in the address map, to match the msi64_translator_page signal value that is independent of the GICD address map reset.
  The msi64_translator_page and its_transr_page_offset signals, or one of either, must not be on top of any other GIC register page.

  To ensure that this method of mapping is hidden from software, all accesses to the GITS_TRANSLATER register must pass through an Encapsulator, or similar embedded functionality. See 3.4 MSI-64 Encapsulator on page 47 for more information.

  The following figure shows an example of how to integrate the MSI-64 Encapsulator in a system. The MSI-64 Encapsulator connects upstream of the interconnect and targets an ITS downstream of the interconnect. In this scenario, the DeviceID is transported on the data channels of the interconnect to the ITS. This topology benefits those systems where the width of the awuser signal on the interconnect is too narrow to transport the DeviceID.

Figure 4-4: MSI-64 Encapsulator with DeviceID sent in the data[63:32] bits



4.12.3 ITS commands and errors

Each ITS detects a wide range of command errors and translation errors, and reports them in Armv8.2 RAS-compliant error records in the Distributor.

The ITS record error syndromes comprise four groups that each have separate enables in the GITS_FCTLR register. The following table shows the ITS record error syndrome groups.

Table 4-3: ITS record error syndrome groups

Group	Control
ACE5-Lite subordinate interface write translation errors. Only when the ITS has a separate ACE5-Lite subordinate port.	GITS_FCTLR.AEE (Access Error Enable)
Translation errors on incoming writes to GITS_TRANSLATER	GITS_FCTLR.UEE (Unmapped Error Enable)
Errors during commands	GITS_FCTLR.CEE (Command Error Enable)

Group	Control
Other errors such as memory system, or memory allocation errors	None

See 4.17.4.15 ITS command and translation error records 27+ on page 114 for information about all the detected syndromes.

ITS commands must be written by software before they are executed.

The ITS Command queue operates a stall mechanism on any error, irrespective of the GITS_FCTLR.CEE value. To execute commands, software writes to a Command queue in memory and then updates the GITS_CWRITER.Offset to indicate that there are commands to run.

- Normally, the GITS_CREADR.Offset increments until it matches the GITS_CWRITER.Offset, wrapping as necessary, to indicate that the Command queue has completed.
- If an error occurs, GITS_CREADR.Stalled is set, which indicates that processing has stopped and software intervention is required. If GITS_FCTLR.CEE is set, at least one error is reported in the relevant error record to aid software debug. You can correct the command that the GITS_CREADR identifies and then resume the Command queue, by writing to GITS_CWRITER.Retry. If the command is no longer required, you must rewrite it as a SYNC command before you resume.

To determine when Command queue execution completes, you can either:

- Poll GITS_CREADR.Offset until it matches GITS_CWRITER.Offset.
- Put an INT command in the queue and wait for that interrupt to arrive.

If you add an INT command, then we recommend that you enable GITS_FCTLR.CEE and that you configure the fault handling interrupt or error recovery interrupt to be delivered to a core that can resolve Command queue issues. See 4.17.3 Error recovery and fault handling interrupts on page 98 for more information.

4.13 LPIs

Locality-specific Peripheral Interrupts (LPIs) are always message-based, and can be from a peripheral, or from a PCIe root complex.

An LPI targets only one core. LPIs are generated when the peripheral writes to the ITS. The ITS contains the registers to control the generation and maintenance of LPIs. The ITS provides INTID translation, allowing peripherals to be owned directly by a virtual machine if an SMMU is also present for those peripherals.

If you use GIC architecture version 3, the ITS enables interrupts to be translated to the ID space of the hypervisor instead of directly to a virtual machine. If you use GIC architecture version 4.1, the hypervisor can configure the ITS to directly send interrupts.

4.13.1 LPI programming and generation

Only an ITS can generate an LPI. See *Locality-Specific Peripheral Interrupts, Arm® Generic Interrupt Controller v3 and v4* for more information.

4.13.2 LPI direct injection

The ITS can directly inject an LPI to a vPE, if the LPI is mapped to a vPE and the ITS uses a `VMAPI` or `VMAPTI` command.

4.13.3 LPI multichip operation

The GIC-720AE does not use physical target addresses, so GITS_TYPER.PTA == 0. Therefore, GIC-720AE uses the value of GICR_TYPER.ProcessorNumber to route all LPIs and commands to their targets.

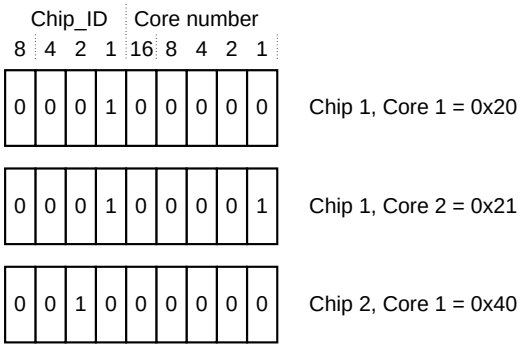
The GIC-720AE splits the GITS_TYPER.ProcessorNumber value into two fields, Chip_ID and the padded linear on-chip core number.

The width of the padded on-chip core number field is defined by the `max_pe_on_chip` configuration parameter. This parameter sets the maximum number of cores or threads on a single chip in the configuration. The width of the linear on-chip core number field is discoverable through GICD_CFGID.PEW.

For example, if `max_pe_on_chip` = 17, the width of the lower part of the on-chip core number field is ceil[$\log_2(17)$] = 5 bits. Therefore, the ProcessorNumber value of the first core on chip 1 is `0x20`, the value of the second core on chip 1 is `0x21`, the value of the first core on chip 2 is `0x40`.

The following figure shows the ProcessorNumber fields with typical values.

Figure 4-5: ProcessorNumber fields



If software attempts to access a chip that does not exist, is offline, or access a core that does not exist, the request is dropped and reported through the ITS command and translation error records.

4.13.4 LPI caching

If LPI support is configured, the GIC-720AE supports a single LPI cache for each chip with up to 4 banks.

The LPI cache is 2-way set associative based on the lowest bits of the LPI INTID, and stores LPI properties from the LPI Property table. The relevant set is checked for valid properties as each LPI arrives in the system. If multiple banks are selected, then to select the bank the GIC uses the lower bits of the core or vPE number.

The cache is fully associative for pending LPIs, which means that the LPI system fills almost all lines in the cache before sending anything to the Pending tables. The GIC-720AE is not optimized for collating LPIs that have the same INTID. However the system is designed to reorder and sort the cache over time. In some circumstances, this behavior can cause duplicated interrupts to not be collated efficiently. However, the reduced use of the Pending table, results in better latency bounds under load.

This method of caching means that priorities are associated with an incoming LPI and remain with it until it is serviced. The GIC does not accept changes in the LPI Property table, until the relevant `INV` and `SYNC` commands are executed through an ITS, GICR_INVLPIR, or GICR_INVALLR.

Up to 16 concurrent `INV` commands can be run at a time. The GIC sets GICR_SYNCR.BUSY to zero as soon as the `INV` is hazarded and any matching interrupts have been recalled from the target PE.

The command slot becomes free after the GIC discovers that the interrupt does not exist in the cache, which might require a linear search depending on load and cache contents. If the cache overflows, then it might also be necessary to check the Pending table for the invalidated ID.

The GIC-720AE considers priority and enable when choosing data to retain in the cache. However, pending interrupts always take priority over interrupts that are not pending, so there is no guarantee that the highest priority interrupt data always remains stored in the cache.

See the GICD_UTILR register for information about how software can use the utilization engines to optimize the LPI cache contents.

Related information

Distributor configuration on page 35

4.13.5 Choosing between LPIs and SPIs

Message-based interrupts can be either LPIs or SPIs.

The decision by software to use an LPI or SPI for an interrupt, depends on whether there are message-based SPIs available and if the GIC-720AE has LPI support. The allocation of message-based SPIs is set during the GIC configuration process. Also, if the hardware ensures that an spi signal is held to a logic level that represents the inactive state, then software can use that SPI ID as a message-based SPI.

The interrupt type can be selected by either making the peripheral write to a different GIC-720AE address, or by changing the address translation for the interrupt write in the SMMU. Changing only the SMMU is possible because the registers for Non-secure message-based interrupts, GICD_SETSPI_NSR, and GITS_TRANSLATER, are at the same address offset in different pages.

The following factors can help you to decide which interrupt type is most appropriate:

- Only the ITS provides INTID translation, therefore LPIs are preferable for peripherals that a virtual machine owns. This is because the hypervisor can let the virtual machine program the peripheral directly, and the ITS converts the virtual machine interrupt IDs to unique physical IDs.
- In GIC architecture version 4.1, the hypervisor can route LPIs directly to a virtual machine. However, SPIs that target a virtual machine, interrupt the hypervisor and are inserted through the CPU interface list registers.
- LPIs are always Group 1 Non-secure, so message-based interrupts that target Secure software must use SPIs.
- Only SPIs are able to target all cores, which means that the GIC-720AE attempts to automatically balance the interrupt load to cores that are active but not handling other interrupts.
- The GIC-720AE can provide more LPIs than SPIs.
- You might decide not to include LPI support in a small system where the features of the ITS are not required and there are few message-based interrupts.
- SPIs usually have a better worst-case interrupt latency than LPIs. This difference is because SPIs have all their settings stored internally to the GIC-720AE, whereas LPIs that are not cached require external memory accesses. The cache hit rate is expected to be higher for the LPIs that occur more frequently. Therefore, we recommend using SPIs for any latency-sensitive interrupts that are expected to occur infrequently.

For more information, see the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4*.

4.13.6 LPI error recovery procedure

Uncorrectable LPI errors can occur in either the LPI cache or the TGT cache. In both cases, the GIC reports the error in the GICT_ERR10MISC1 register and normal operation continues.

When an uncorrectable error occurs, the GICT_ERR<n>MISC1 register contains the RAM contents of the corrupted line. The line in RAM is dropped, and any pending interrupts that it might contain are lost.

Software can use the data in the GICT_ERR<n>MISC1 register to check several interrupt sources, such as the corrupted INTID. This ID is never more than 2 bits away from the recorded ID. In this case, no recovery in the GIC is required, other than to clear the error record.

Software must decide whether to abort, check interrupt sources, or continue with the expectation that the interrupt source repeats the LPI.

4.14 Memory access and attributes

The LPI and ITS translations and properties are located in memory tables whose locations are defined in registers that specify their base address, size, and access attributes.

We recommend that all tables are placed in Normal memory. All ITS tables are private, and after allocation, only the GIC accesses them. However, the LPI Property table and ITS Command queue are written to by cores, and read by the GIC.

The following table shows the registers that configure accesses to the LPI and ITS tables. The mappings are designed for the Armv8, Armv8.2, and Armv8.4 generation of cores. However, setting the relevant DCC bits converts the GIC-720AE to full featured mapping.

Table 4-4: Memory access registers

Access type	Register	Mapping control bit
LPI Property table	GICR_PROPBASER	GICD_FCTLR2.DCC
LPI Pending table	GICR_PENDBASER	
LPI virtual Property tables	ITS `VMAPP` command	
LPI virtual Pending tables		
vPE Configuration table	GICR_VPROPBASER and GITS_BASER2	
ITS Device table	GITS_BASER0	GITS_FCTLR.DCC
ITS translation table	GITS_BASER0	
ITS Collection table	GITS_BASER1	
ITS Command queue	GITS_CBASER	

The main Cacheability value is derived from the *BASER*.OuterCache field, unless it is zero, in which case the Cacheability value is a value that the following table shows.

Table 4-5: Cacheability values

Main Cacheability value (*BASER*.OuterCache)	Other Cacheability value (*BASER*.InnerCache)	arcache signal	awcache signal	arcache signal (DCC = 1)	awcache signal (DCC = 1)
`0b000`, Device-nGnRnE	-	`0b0010`	`0b0010`	`0b0010`	`0b0010`
`0b001`, Normal Non-cacheable	Match	`0b0011`	`0b0011`	`0b0011`	`0b0011`
	No match	`0b0011`	`0b0011`	`0b0011`	`0b0011`
`0b010`, Normal Cacheable RA Write-Through	Match	`0b0011`	`0b0011`	`0b1110`	`0b0110`
	No match	`0b0011`	`0b0011`	`0b1110`	`0b0110`
`0b011`, Normal Cacheable RA Write-Back	Match	`0b1111`	`0b0111`	`0b1111`	`0b0111`
	No match	`0b0011`	`0b0011`	`0b1111`	`0b0111`
`0b100`, Normal Cacheable WA Write-Through	Match	`0b0011`	`0b0011`	`0b1010`	`0b1110`
	No match	`0b0011`	`0b0011`	`0b1010`	`0b1110`
`0b101`, Normal Cacheable WA Write-Back	Match	`0b1011`	`0b1111`	`0b1011`	`0b1111`
	No match	`0b0011`	`0b0011`	`0b1011`	`0b1111`

Main Cacheability value (*BASER*.OuterCache)	Other Cacheability value (*BASER*.InnerCache)	arcache signal	awcache signal	arcache signal (DCC = 1)	awcache signal (DCC = 1)
0b110, Normal Cacheable WA RA Write-Through	Match	0b0011	0b0011	0b1110	0b1110
	No match	0b0011	0b0011	0b1110	0b1110
0b111, Normal Cacheable WA RA Write-Back	Match	0b1111	0b1111	0b1111	0b1111
	No match	0b0011	0b0011	0b1111	0b1111

The a<x>domain signal is driven according to the *BASER*.Shareability field unless the resultant Cacheability is Device or Non-cacheable. For Device or Non-cacheable accesses, a<x>domain becomes 0b11, that is, system Shareable in accordance with the *AMBA® AXI Protocol Specification*.

4.14.1 MPAM information

The GIC-720AE supports *Memory Partitioning and Monitoring* (MPAM) and it assigns PARTIDR and PMG values to all memory accesses that it issues on the ACE5-Lite manager interface.

There is one copy of GICR_PARTIDR for all cores on a chip, so the cores must all use the same value.

GICR_PARTIDR is used for all accesses apart from ITS tables that use GITS_PARTIDR.

Accesses to the vPE Configuration table use either GICR_PARTIDR when at least one GICR_VPROPBASER is valid, or alternatively the GITS_BASER of the issuing ITS.

MPAM has no effect on cache allocation or partitioning within the GIC.

4.15 Power management

The GIC-720AE can be powered down by the system power controller. The GIC also supports the power controller powering down the cores that the GIC services. The GICR_WAKER and the GICR_PWRR registers provide bits to control functions that are associated with power management.

4.15.1 Redistributor power management

At reset, the Redistributors are considered to be powered down. To power up the Redistributors, software must use the GICR_PWRR register.

Note

This requirement is true for all GIC-720AE configurations.

The GICR_PWRR register can control Redistributor power management either by operating through the core, or through the Redistributor.

If operating through the core, each core must program its GICR_PWRR.RDPD = 0 and GICR_PWRR.RDAG = 0 to ensure that the Redistributor powers up. Alternatively, a single core can power up the Redistributor for all cores that connect to the same Redistributor by writing GICR_PWRR.RDPD = 0 and GICR_PWRR.RDAG = 1.

You can use GICR_PWRR.RDG to identify which core shares a Redistributor.

The powerdown sequence is shown in the following pseudocode:

```
Power off (setting RDPD to 1):
    // Check group not transitioning.
    repeat
    until (GICR_PWRR.RDGPD == GICR_PWRR.RDGPO)

    // Write to power the CPU off.
    GICR_PWRR.RDPD = 1;
```

The powerup sequence is shown in the following pseudocode:

```
Power on (setting RDPD to 0):
    repeat
        // Check group not transitioning.
        repeat
        until (GICR_PWRR.RDGPD == GICR_PWRR.RDGPO)

        // Write to power the CPU on.
        GICR_PWRR.RDPD = 0;

        // Check access, if RDPD == 0 then powered on.
    until (GICR_PWRR.RDPD == 0)
```

GICR_PWRR must be accessed using the GICR address space that relates to the core being powered on or off.

Some GICR_* registers are not accessible until software programs GICR_PWRR.

4.15.2 Processor core power management

The GIC architecture defines the programming sequence to safely power down a core that connects to the GIC-720AE.

The powerdown programming sequence uses the GICR_WAKER.ProcessorSleep bit. When all cores within a cluster are powered down using the architectural sequence, you can power gate the GIC Stream interface for that cluster.

Before a core is powered down, you must set the GICR_WAKER.ProcessorSleep bit to 1. The core must then poll the GICR_WAKER.ChildrenAsleep bit to ensure that there are no outstanding transactions on the GIC Stream interface of the core.

To ensure that there are no interrupts during the powerdown of the core, in a typical powerdown sequence you must:

1. Mask interrupts on the core.
2. Clear the CPU interface enables.
3. Set the interrupt bypass disable on the CPU interface.


Note

The core powerdown sequence that you use must match the core powerdown sequence that is described in the Technical Reference Manual for your processor.

When a core is powered down and the GICR_WAKER.ProcessorSleep bit is set to 1, if the GIC-720AE receives an interrupt that targets only that core, the Wake Request block asserts the wake_request signal that corresponds to that core. The wake_request signal must connect to the system power controller. See 3.6 Wake Request on page 53.

You must not set the GICR_WAKER.ProcessorSleep bit to 1, unless the core enters a power state where the GIC-720AE uses a power controller to wake the core instead of the GIC Stream interface. For example, with Armv8 and Armv9 processors, if a core enters a low-power state that is based on the *Wait For Interrupt* (`WFI`) or *Wait For Event* (`WFE`) instructions, such as retention, you must not set the GICR_WAKER.ProcessorSleep bit to 1.

Interrupts can cause the core to leave the low-power state, entered by executing a `WFI` or `WFE` instruction, as defined in the *Arm® Architecture Reference Manual for A-profile architecture*. The system integrator can use a cpu_active signal to ensure that interrupts that can target multiple cores are much less likely to target cores in certain low-power states. In such a system, software has more control of the conditions under which cores leave low-power states.

Interrupts that target only one core are unaffected by the cpu_active signal and are always sent to that core. Also, if the GICR_WAKER.ProcessorSleep bit for that core is set, the wake_request signal is asserted for that core.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for information about power management, and about wakeup signals and their relation to the core outputs.

4.15.3 Power control and P-Channel

You can use the P-Channel to isolate a chip from the system.

The P-Channel has the following states:

RUN (pstate signal == 0x0)
: The normal functional mode

CONFIG (pstate signal == 0x9)
: The GIC does not send any cross-chip messages. It accepts incoming messages but does not process them.

OFF (pstate signal == `0xF`)

The GIC does not send any cross-chip messages and does not accept any incoming messages. The icrdtready signal is clamped LOW to prevent accesses entering the GIC.

While in both the CONFIG and OFF states, register accesses that are normally sent to another chip are serviced locally. Therefore, the Routing table registers read the local versions instead of the copies of the Routing table owner. The same is true for SPIs that are owned remotely. Therefore, it is safe to save and restore the Distributor register values in either of these P-Channel states.

The GIC can exit reset in either the RUN or OFF states by setting the initial value of the pstate signal. If you have saved register values and intend to restore them, you must use the OFF state and restore the Routing table first, before attempting to restore any SPI registers.

Related information

Isolating a chip from the system on page 370

4.15.4 SPI RAM retention

If the GIC is regularly powered down and reset, then saving and restoring state can be time consuming when there are many SPIs. At the exit of reset, the spi_ram_retained signal enables the GIC to trust the SPI programming state that the RAMs contain.

The `u_spi_ram0` and `u_spi_ram1` instances store the state of the following registers:

- GICD_IGROUPRn(E)
- GICD_ISENABLERn(E)
- GICD_ISPENDRn(E)
- GICD_ISACTIVERn(E)
- GICD_IPRIORITYRn(E)
- GICD_ICFGRn(E)
- GICD_IGRPMODRn(E)
- GICD_NSACRn(E)
- GICD_IROUTERn(E)
- GICD_I*ERR*
- GICD_IVIEWRn(E), when the GIC supports multi view.

If the spi_ram_retained signal is HIGH when GIC-720AE exits reset, then the GICD trusts the data in the `u_spi_ram0` and `u_spi_ram1` RAM instances.

The GIC-720AE does not drive the RAM retention signals. Therefore, after the GIC enters the Q_STOPPED state, some other logic must drive the RAM retention signals.

Entering SPI RAM retention

To prepare for SPI RAM retention, perform the following steps:

1. Ensure that the GIC is IDLE by:
   a. Completing the GICR_WAKER.ProcessorSleep handshake for all PEs.
   b. Clear the GICD_CTLR.Enable* bits for all groups and then poll for GICD_CTLR.RWP == 0.
2. Put all SPI Collators into the Q_STOPPED state.
   This step ensures that no more interrupts can arrive from the SPI Collators. The SPI Collators deny any interrupts that are not in their idle state. Depending on the system configuration, the system might need to include some external interrupt masking.
3. Put the GIC into the Q_STOPPED state by using the GICD Q-Channel interface.
4. The system drives the necessary signals to the SPI RAMs, to put them into retention.
   The system integrator could use the RAM I/O sideband signals to connect the retention signals to the RAM instances.

The system can now safely power down the GIC.

Restoring from SPI RAM retention

To restore from SPI RAM retention, perform the following steps:

1. Power up the GIC.
2. Drive the spi_ram_retained signal HIGH before exiting reset.
   Only change the state of the spi_ram_retained signal while in reset.
3. Exit reset on the GICD.
4. Reprogram the GIC as normal, except that all registers that scale with SPIs retain their values. Software must program all Redistributor registers (GICR) and common non-scaling registers such as GICD_CTLR.
5. Put the GIC into the Q_RUN state, by using the GICD Q-Channel interface.
6. Put all SPI Collators into the Q_RUN state.
7. Set the GICD_CTLR.Enable* bits for all groups and then poll for GICD_CTLR.RWP == 0.
8. Wake all the PEs.

4.16 Performance Monitoring Unit

The GIC-720AE contains a PMU for counting the main GIC events from the Distributor and any configured ITS blocks on the same chip.


Note

The PMU does not track *GIC Cluster Interface* (GCI) events. Software can count the delivery of PPI and SGI interrupts by recording calls to the core interrupt service routine.

The GIC events are described in Table 5-116: GICP_EVTYPERn.EVENT field encoding on page 272.

The PMU has five counters with snapshot capability and overflow interrupt.

Secure and Non-secure interrupts are counted together so Non-secure software cannot, by default, access the GICP (PMU) register space. However, Secure software can decide to allow access. Non-secure software can be given access to the GICP (PMU) register space by either:

- Software programming the GICD_SAC.GICPNS bit to 1.
- Setting the gicp_allow_ns tie-off signal HIGH, during silicon integration.

If GICD_CTLR.DS == 1, the GICP register space is accessible to all software.

PMU and multi view support

If a GIC configuration supports multi view, the GICD_SAC.GICPVIEW bit controls which view can access the GICP page and the events that the PMU records.

When GICD_SAC.GICPVIEW is 0, the PMU records events from all views. The GIC allows only view 0 to access the GICP page.

When GICD_SAC.GICPVIEW is nonzero, the PMU records events only from the programmed view. The GIC allows only view 0 and the programmed view to access the GICP page.

Overflow interrupt

Software can enable the overflow interrupt for each counter by setting the relevant bit of GICP_INTENSET0. For example, bit[0] enables GICP_EVCNTR0 and bit[1] enables GICP_EVCNTR1. Similarly, software can disable the overflow interrupt enable by corresponding writes to GICP_INTENCLR0.

When enabled, the interrupt activates at any of these events:

- A write to a GICP_OVSSET0 for any counter.
- An overflow on any enabled counter.

The GICP_OVSSET0 and GICP_OVSCLR0 registers can be used for save and restore operations and for testing the correct integration of the pmu_int interrupt signal.

The pmu_int signal can be used to trigger external logic, for example, to trigger a read of the captured data.

Alternatively, by programming a valid SPI ID into the GICP_IRQCR.SPIID field, the pmu_int signal SPI is delivered internally in accordance with normal SPI programming.

If software observes excessive assertion of the pmu_int signal, then the software must prevent pmu_int from interfering with mission operation. For example, if the PMU causes an SPI ID to be delivered internally, software must disable the corresponding SPI ID.

The GICP_IRQCR.SPIID field must be programmed to 0 if internal routing is not required, or if internal routing is required, to a legally supported SPI ID. If the programmed ID value is less than 32 or out of range, or for multichip configurations, not owned on chip, the register updates to 0 and no internal delivery occurs.

Snapshot

Each PMU counter GICP_EVCNTRn has a corresponding GICP_SVRn snapshot register. On a snapshot event, all five counters are copied to their backup registers so that all consistent data is copied out over a longer period.

The snapshot events are:

- A handshake on the 4-phase sample_req/sample_ack signal external handshake.
- A write of 1 to the GICP_CAPR.CAPTURE bit.
- An overflow of an enabled counter when GICP_EVTYPERn.OVFCAP is set.

There is one set only of snapshot registers, so data is replaced in multiple capture events.

4.17 Reliability, Accessibility, and Serviceability

The GIC-720AE uses a range of RAS features for all RAMs, which include *Single Error Correction and Double Error Detection* (SECDED), and Scrub, software and bus error reporting.

The GIC makes all necessary information available to software through Armv8.2 RAS architecture-compliant register space.

4.17.1 Non-secure access

You can control whether Non-secure software has access to the RAS architecture-compliant register space by using GICD_SAC.GICTNS. The gict_allow_ns tie-off signal sets the reset value of the GICTNS bit.

If there is an error, and if GICD_CTLR.DS == 0, all SPIs, PPIs, and SGIs resort to a Secure group. Therefore, interrupt programming is not revealed to the Non-secure side.

4.17.2 Error record classification

The GIC reports errors in Armv8.2 RAS architecture-compliant error records, which are accessible through the ACE5-Lite subordinate programming interface.

The classes of error records are:

- Correctable ECC errors.
- Uncorrectable ECC errors.
- ITS command and translation errors.
- Software access errors.

The error records have a separate reset so that they can be read after a main GIC reset to determine any problems.

4.17.3 Error recovery and fault handling interrupts

You can assign a recorded correctable ECC error to the fault handling interrupt by setting GICT_ERR<n>CTLR.CFI.

All correctable ECC errors have error counters, but the interrupt fires on every error.

You can assign a recorded uncorrectable ECC error either to the fault handling interrupt, fault_int signal, by setting GICT_ERR<n>CTLR.FI, or to the error recovery interrupt, err_int signal, by setting GICT_ERR<n>CTLR.UI. The interrupt fires on every uncorrectable interrupt occurrence irrespective of the counter value.

You can route the fault_int and err_int signals out as interrupt wires for situations where error recovery is handled by a core that does not receive interrupts directly from the GIC, such as a central system control processor. Alternatively, you can drive each interrupt internally by programming the associated GICT_ERRIRQCR<n> register.

Each GICT_ERRIRQCR<n> register contains an ID field that must be programmed to 0 if internal routing is not required, or if internal routing is required, to a legally supported SPI ID. If the programmed ID value is less than 32, out of range, or not owned on chip for multichip configurations, the register updates to 0 and no internal delivery occurs.

We recommend that if the err_int and fault_int signals are internally routed, the target interrupts must not have SPI Collator wires, or if they are present they are tied off. This recommendation prevents software checking for the same ID at multiple destinations.

The err_int and fault_int signals do not have direct test enable registers. You can test connectivity using error record 0 and triggering an error, such as an illegal AXI access to a nonexistent register.

4.17.4 Error handling records

The GIC-720AE has several error records. The range of error handling records that are available depends on the configuration of the GIC-720AE.

If a GIC configuration supports multi view, the RAM error records are visible only to view 0.

The following table lists the GIC-720AE error handling records. The Type column uses the following acronyms:

CE Correctable error
UEO Restartable error and contained
UER Recoverable error

Table 4-6: Error handling records

Record	Description	Type	Description, events, and recovery sequences
0	Software error in GICD programming	UEO	Table 4-7: Software errors, record 0 on page 100

Record	Description	Type	Description, events, and recovery sequences
1	Correctable SPI RAM errors	CE	Table 4-8: SPI RAM errors, records 1-2 on page 106. GICT_ERR<n>STATUS.SERR == 7, data value from associative memory.
2	Uncorrectable SPI RAM errors	UER	
3	Correctable SGI RAM errors	CE	Table 4-9: SGI RAM errors, records 3-4 on page 107. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
4	Uncorrectable SGI RAM errors	UER	
5	Correctable TGT_SPI cache errors	CE	Table 4-10: TGT_SPI RAM errors, records 5-6 on page 107. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
6	Uncorrectable TGT_SPI cache errors	UER	
7	Correctable PPI RAM errors	CE	Table 4-11: PPI RAM errors, records 7-8 on page 108. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
8	Uncorrectable PPI RAM errors	UER	
9	Correctable LPI RAM errors	CE	Table 4-12: LPI RAM errors, records 9-10 on page 109. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory. Records 9-10 are not present if there is no LPI support.
10	Uncorrectable LPI RAM errors	UER	
11	Correctable PTS RAM errors	CE	4.17.4.7 PTS RAM error records 11-12 on page 109. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
12	Uncorrectable PTS RAM errors	UER	
13	Correctable TGT_LPI RAM errors	CE	4.17.4.8 TGT_LPI RAM error records 13-14 on page 109. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
14	Uncorrectable TGT_LPI RAM errors	UER	
15	Correctable VICM RAM errors	CE	4.17.4.9 VICM RAM error records 15-16 on page 110. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory. Records 15-24 are not present if there is no vLPI support.
16	Uncorrectable VICM RAM errors	UER	
17	Correctable VSPA RAM errors	CE	4.17.4.10 VSPA RAM error records 17-18 on page 110. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
18	Uncorrectable VSPA RAM errors	UER	
19	Correctable VTGT_VSTR RAM errors	CE	4.17.4.11 VTGT_VSTR RAM error records 19-20 on page 111. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
20	Uncorrectable VTGT_VSTR RAM errors	UER	
21	Correctable VTGT_VRES RAM errors	CE	4.17.4.12 VTGT_VRES RAM error records 21-22 on page 112. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
22	Uncorrectable VTGT_VRES RAM errors	UER	
23	Correctable VTGT_SRCH RAM errors	CE	4.17.4.13 VTGT_SRCH RAM error records 23-24 on page 112. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
24	Uncorrectable VTGT_SRCH RAM errors	UER	
25	Correctable error from ITS RAM	CE	Table 4-20: ITS RAM errors, records 25-26 on page 113. GICT_ERR<n>STATUS.SERR == 6, data value from associative memory.
26	Uncorrectable error from ITS RAM	UEO	
27-27+`ITS_NUM`	ITS command and translation errors	UER	4.17.4.15 ITS command and translation error records 27+ on page 114. GICT_ERR<n>STATUS.SERR == 14, illegal access. One record for each ITS on the chip.
62	Correctable error from CC RAM	CE	4.17.4.16 CC RAM error records 62-63 on page 129
63	Uncorrectable error from CC RAM	UEO	

4.17.4.1 Software error record 0

Software error record 0 records software errors that are uncorrectable.

Record 0 contains software programming errors from a wide range of sources within the GIC-720AE. In general, these errors are contained. For uncorrected errors, the information that is provided gives enough information to enable recovery without significant loss of functionality.

We recommend that record 0 is connected to a high priority interrupt. This connection prevents the record from overflowing if it receives more errors than it is able to process with the possible loss of information that is required for recovery. See 4.17.3 Error recovery and fault handling interrupts on page 98 for more information.

The following table describes the syndromes that are recorded in record 0, the reported information, and recovery instructions.

Table 4-7: Software errors, record 0

GICT_ERR<n>STATUS.IERR (Syndrome)	GICT_ERR<n>STATUS .SERR	GICT_ERR<n>MISC0. Data description (other bits RES0) Always packed from 0 (lowest = 0)	Recovery, prevention
0x0, SYN_ACE_BAD Illegal ACE5-Lite subordinate access.	0xE	AccessRnW, bit[12] AccessSparse, bit[11] AccessSize, bits[10:8] AccessLength, bits[7:0]	Repeat illegal access, with appropriate size and properties. Full access address is given in GICT_ERR0ADDR.
0x1, SYN_PPI_PWRDWN Attempt to access a powered down Redistributor.	0xF	Redistributor, bits[24:16] Core, bits[8:0]	Ensure that the Redistributor is powered up before accessing. See GICR_PWRR. Attempt was made by the core reported in MISC0.
0x2, SYN_PPI_PWRCHANGE Attempt to power down Redistributor rejected.	0xF	Redistributor, bits[24:16] Core, bits[8:0]	Ensure that the core accessing the register, or all cores with the same GICR_PWRR.RDG if GICR_PWRR.RDAG is set, has completed the GICR_WAKER.ProcessorSleep handshake.
0x4, SYN_PROPBASE_ACC Attempt to reprogram PROPBASE registers to a value that is not accepted because another value is already in use.	0xF	Core, bits[8:0]	GICR_PROPBASER is shared between all cores on a chip. When any GICR_CTLR.Enable_LPIs bit is set, the value is locked and cannot be updated unless a complete GICR_WAKER.Sleep handshake is complete. See A.2 Other power management on page 364.
0x5, SYN_PENDBASE_ACC Attempt to reprogram PENDBASE registers to a value that is not accepted because another value is already in use.	0xF	Core, bits[8:0]	When any GICR_CTLR.Enable_LPIs bit is set, the Shareability, InnerCache, and OuterCache fields are locked for the whole chip. They can only be changed by completing the GICR_WAKER.Sleep handshake. See A.2 Other power management on page 364. Otherwise, repeat the register access using the current global values.

GICT_ERR<n>STATUS.IERR (Syndrome)	GICT_ERR<n>STATUS .SERR	GICT_ERR<n>MISC0. Data description (other bits RES0) Always packed from 0 (lowest = 0)	Recovery, prevention
0x7, SYN_WAKER_CHANGE Attempt to change GICR_WAKER abandoned due to handshake rules.	0xF	Core, bits[8:0]	GICR_WAKER.ProcessorSleep and GICR_WAKER.ChildrenAsleep form a 4-phase handshake. The attempt to change state must be repeated when the previous transition has completed.
0x8, SYN_SLEEP_FAIL Attempt to put GIC to sleep failed as cores are not fully asleep.	0xF	Core, bits[8:0]	All cores must be asleep, using the GICR_WAKER.ProcessorSleep handshake, before you flush the LPI cache using GICR_WAKER.Sleep.
0x9, SYN_PGE_ON_QUIESCE Core put to sleep before its Group enables were cleared.	0xF	Core, bits[8:0]	The core must disable its group enables before it toggles the GICR_WAKER.ProcessorSleep handshake, otherwise, the GIC clears its record of the Group enables, causing a mismatch between the GIC and the core.
0x10, SYN_SGI_NO_TGT SGI sent with no valid destinations.	0xE	Core, bits[8:0]	If the SGI is required, software must repeat the SGI from the reported core with a valid target list. If this level of RAS functionality is required, the software must track generated SGIs externally.
0x11, SYN_SGI_CORRUPTED SGI corrupted without effect.	0x6	Core, bits[8:0]	An SGI is corrupted due to a RAM error in the PPI RAM. The RAM error details are reported separately in record 8. The GIC ignores the SGI generated from the recorded core. If you want software to recover from this error, it must use an external record of the generated SGI.
0x12, SYN_GICR_CORRUPTED Data was read from GICR register space that has encountered an uncorrectable error.	0x6	GICT_ERR0ADDR is populated	Software has tried to read corrupted data that is stored in SGI RAM or PPI RAM. Check records 4 and 8, and perform a recovery sequence for those interrupts.
0x13, SYN_GICD_CORRUPTED Data was read from GICD register space that encountered an uncorrectable error.	0x6	GICT_ERR0ADDR is populated	Software has tried to read corrupted data that is stored in SPI RAM. Check record 2 and perform a recovery sequence for those interrupts.
0x14, SYN_ITS_OFF Data was read from an ITS that is powered down.	0xF	GICT_ERR0ADDR is populated	Ensure that the qreqn_its<x> signal power control Q-Channel is in the RUN state before accessing the relevant ITS.
0x18, SYN_SPI_BLOCK Attempt to access an SPI block that is not implemented.	0xE	Block, bits[4:0]	No recovery is required. Correct the software.
0x19, SYN_SPI_OOR Attempt to access a non-implemented SPI using (SET|CLR)SPI.	0xE	ID, bits[9:0]	Reprogram the issuing device so that it sends a supported SPI ID.
0x1A, SYN_SPI_NO_DEST_TGT An SPI has no legal target destinations.	0xF	ID, bits[9:0]	Before enabling the specified SPI, reprogram the SPI to target an existing core. The same SPI might repeat this error several times and cause an overflow.

GICT_ERR<n>STATUS.IERR (Syndrome)	GICT_ERR<n>STATUS .SERR	GICT_ERR<n>MISC0. Data description (other bits RES0) Always packed from 0 (lowest = 0)	Recovery, prevention
0x1B, SYN_SPI_NO_DEST_1OFN A 1 of N SPI cannot be delivered due to bad GICR_CTLR.DPG<0|1NS|1S> or GICR_CLASSR programming.	0xF	ID, bits[9:0]	Ensure that there is at least one valid target for the specified 1 of N interrupt, that is, ensure that at least one core has acceptable DPG and CLASS settings to enable delivery. The same SPI might repeat this error several times and cause an overflow.
0x1C, SYN_COL_OOR A collator message is received for a non-implemented SPI, or is larger than the number of owned SPIs in a multichip configuration.	0xF	ID, bits[9:0]	In a multichip configuration, ensure that there are enough owned SPIs to support all SPI wires that are used. Any unsupported interrupts must be disabled at the source.
0x1D, SYN_DEACT_IN A `Deactivate` command to a nonexistent SPI, or with incorrect groups set. `Deactivate` commands to LPI and nonexistent PPI are not reported.	0xE	None	A `Deactivate` command occurred to a nonexistent SPI, or that SPI group prevents the deactivate occurring. Software must check the active states of SPIs.
0x25, SYN_VSGI_OFFLINE Pending vSGI to a vPEID mapped to an offline chip.	0xF	Chip [$\log_2$(chips)−1:0] ID (multi-hot) [15:0] vPEID[$\log_2$(vpes)−1:0]	Software must ensure that vPEs are either moved off chips or unmapped, before it takes the chip offline.
0x30, SYN_VSGI_UNMAPPED Pending vSGI to a vPEID that is not mapped.	0xF	ID (multi-hot) [15:0] vPEID[$\log_2$(vpes)−1:0]	Software must not attempt to generate vSGIs to unmapped vPEs.
0x33, SYN_VSGI_LOST Pending vSGI to a vPEID that has inconsistent mapping information across multiple chips.	0xF	ID (multi-hot) [15:0] vPEID [$\log_2$(vpes)−1:0]	Software must check for any reported uncorrectable errors. Software must also ensure that it issues the correct sequence of `VMAPP(V=xA=x)` commands, as the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* describes.
0x34, SYN_VPT_READ_FAIL An attempt was made to read the vPE configuration from the virtual Pending table, with an error received with the read response.	0x12	vPEID [$\log_2$(vpes)−1:0]	Software must check the memory system and ensure that a valid and accessible address has been provided in the `VMAPP(V1A1)` command.
0x35, SYN_VPT_WRITE_FAIL An attempt was made to write the vPE configuration to the virtual Pending table, with an error received with the write response. The vICM reports bad write responses on the chip where the access occurs, rather than the chip that contains the ITS that generated the command or interrupt.	0x12	vPEID [$\log_2$(vpes)−1:0]	Software must check the memory system and ensure that a valid and accessible address has been provided in the `VMAPP(V1A1)` command.

GICT_ERR<n>STATUS.IERR (Syndrome)	GICT_ERR<n>STATUS .SERR	GICT_ERR<n>MISC0. Data description (other bits RES0) Always packed from 0 (lowest = 0)	Recovery, prevention
0x39, SYN_VPE_CFG_PTR_FAIL An attempt was made to access an indirect vPE Configuration table with an invalid level 2 pointer.	0xD	vPEID [log_2(vpes)−1:0]	Software must ensure that the L1 entries in the vPE Configuration table, point to legal accessible memory.
0x3A, SYN_VPE_CFG_TOP_READ_FAIL An attempt was made to read the level 1 of an indirect vPE Configuration table, with an error received with the read response.	0x12	vPEID [log_2(vpes)−1:0]	Software must ensure that the GITS_BASER2 and GICR_VPROPBASER registers point to legal accessible L1 table when using indirect tables. This memory system error usually results in significant corruption of vPE state, especially if a co-incident vICM RAM error has occurred. Reading GICR_VERRR on all GICDs should indicate corrupted vPEs, assuming that GICR_WAKER.Sleep is not used after generation of this error.
0x3B, SYN_VPE_CFG_LEAF_READ_FAIL An attempt was made to read the level 2 of an indirect vPE Configuration table or any vPE Configuration read when the table is not indirect, with an error received with the read response.	0x12	vPEID [log_2(vpes)−1:0]	Software must ensure that the L1 entries in the vPE Configuration table, point to legal accessible memory. If coincident vICM RAM errors are reported, then the tracking of vPE error state might be lost.
0x3C, SYN_VPE_CFG_WRITE_FAIL An attempt was made to write the level 2 of an indirect vPE Configuration table or any vPE Configuration write when the table is not indirect, with an error received with the write response. The vICM reports bad write responses on the chip where the access occurs, rather than the chip that contains the ITS that generated the command or interrupt.	0x12	vPEID [log_2(vpes)−1:0]	Software must ensure that the L1 entries in the vPE Configuration table, point to legal accessible memory.
0x3D, SYN_VPE_CFG_OVERFLOW A vPE Configuration table access was aborted due to table entry overflow in the address space.	0xD	vPEID [log_2(vpes)−1:0]	Software must not program the vPE Configuration table address to a value that might cause subsequent table accesses to overflow the available memory.
0x40, SYN_LPI_PROP_READ_FAIL An attempt was made to read properties for a single interrupt, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	Software must reprogram the LPI Property table for the specified ID with error-free data and then issue an INV command through the ITS. If an overflow occurred, an INVALL command must be issued to all cores.
0x41, SYN_PT_PROP_READ_FAIL An attempt was made to read properties for a block of interrupts, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	Software must reprogram the LPI Property table for the specified ID with error-free data and then issue an INV command through the ITS. If an overflow occurred, an INVALL command must be issued to all cores.

GICT_ERR<n>STATUS.IERR (Syndrome)	GICT_ERR<n>STATUS.SERR	GICT_ERR<n>MISC0. Data description (other bits RES0) Always packed from 0 (lowest = 0)	Recovery, prevention
0x42, SYN_PT_COARSE_MAP_READ_FAIL An attempt was made to read the coarse map for a target, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16]	No recovery is necessary because the GIC assumes that the coarse map is full.
0x43, SYN_PT_COARSE_MAP_WRITE_FAIL An attempt was made to write the coarse map for a target, with an error received with the write response.	0x12	Virtual, bit[30] Target, bits[29:16]	The GIC attempts to continue, however this error indicates issues with the memory system, and operation might be unpredictable.
0x44, SYN_PT_TABLE_READ_FAIL An attempt was made to read a block of interrupts from a Pending table, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	Software must determine the reason for the pending error read fail. The GIC uses the data that is supplied, however, it is possible for the LPI interrupt to be lost around the specified LPI.
0x45, SYN_PT_TABLE_WRITE_FAIL An attempt was made to write-back a block of interrupts from a Pending table, with an error received with the write response.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	The GIC tries to continue, however, this error indicates issues with the memory system, and operation might be unpredictable.
0x46, SYN_PT_SUB_TABLE_READ_FAIL An attempt was made to read a subblock of interrupts from a Pending table, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	Software must determine the reason for the pending error read fail. The GIC uses the data that is supplied, however, it is possible for the LPI interrupt to be lost around the specified LPI.
0x47, SYN_PT_TABLE_WRITE_FAIL_BYTE An attempt was made to write-back a subblock of interrupts from a Pending table, with an error received with the write response.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	The GIC tries to continue, however, this error indicates issues with the memory system, and operation might be unpredictable.
0x48, SYN_DBL_PROP_READ_FAIL An attempt was made to read properties for a single doorbell, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	Software must ensure that GICR_PROPBASER registers point at a legal accessible LPI property table. The doorbell is cached as disabled so a recovery attempt must issue an INVDB command to the specified vPE.
0x50, SYN_VPROPBASER_DATA An attempt was made to program additional GICR_VPROPBASER.Valid bits with data mismatching GICR_VCFGBASER.	0xF	CPU [$\log_2$(cpus)−1:0]	Software must ensure that the following registers point at the same table in memory: • All GICR_VPROPBASER registers (with GICR_VPROPBASER.Valid == 1). • All GITS_BASER2 registers with corresponding GITS_CTLR.Enabled == 1 of ITS and GICD blocks on the same chip.

GICT_ERR<n>STATUS.IERR (Syndrome)	GICT_ERR<n>STATUS .SERR	GICT_ERR<n>MISC0. Data description (other bits RES0) Always packed from 0 (lowest = 0)	Recovery, prevention
0x52, SYN_VERRR_BUSY An attempt was made to access GICR_VERRR while the register is busy from a previous operation.	0xF	CPU [$\log_2$(cpus)−1:0]	When using GICR_VERRR to extract debug information, then software must ensure that GICR_VERRR.Busy = 0. **Note:** There is one common copy of GICR_VERRR that is shared between all GICR register spaces.
0x53, SYN_VERRR_ALLOC An attempt was made to access GICR_VERRR while there is no vPE Configuration table allocation.	0xF	CPU [$\log_2$(cpus)−1:0]	Before software attempts to use GICR_VERRR, it must ensure that the vPE Configuration table is allocated with either GICR_VPROPBASER.Valid == 1 or GITS_CTLR.Enabled == 1.
0x54, SYN_VERRR_VPE_OOR A request was made to GICR_VERRR with a vPEID that is out of range.	0xE	CPU [$\log_2$(cpus)−1:0]	When using GICR_VERRR, software must only access vPEs within the range that the allocated vPE Configuration table specifies.
0x56, SYN_VSGIR_ALLOC An attempt was made to access GICR_VSGIR while there is no vPE Configuration table allocation.	0xF	CPU [$\log_2$(cpus)−1:0]	Before software attempts to use GICR_VSGIR, it must ensure that the vPE Configuration table has been allocated with either GICR_VPROPBASER or GITS_BASER2.
0x57, SYN_VSGIR_VPE_OOR A request was made to GICR_VSGIR with a vPEID that is out of range.	0xE	CPU [$\log_2$(cpus)−1:0]	When software uses GICD_VSGIR, it must only access vPEs within the range that the allocated vPE Configuration table specifies.
0x58, SYN_VINV_BUSY An attempt was made to access GICR_VINVCHIPR while the register is busy from a previous operation.	0xF	CPU [$\log_2$(cpus)−1:0]	Before software attempts to start an invalidate operation, it must ensure that GICR_VINVCHIPR is not busy by checking that GICR_VINVCHIPR.Valid == 0.
0x59, SYN_VINV_ALLOC An attempt was made to access GICR_VINVCHIPR while there is no vPE Configuration table allocation.	0xF	CPU [$\log_2$(cpus)−1:0]	Before software attempts to use GICR_VINVCHIPR, it must ensure that the vPE Configuration table has been allocated with either GICR_VPROPBASER or GITS_BASER2.
0x70, SYN_ITS_REG_INV_BUSY An attempt was made to invalidate an interrupt while register busy.	0xF	Core, bits[31:16] Data, bits[15:0]	Software must ensure that either or both of the GICR_INVLPI and GICR_INVALLR registers are idle, by checking GICR_SYNCR before, or after, each use.
0x71, SYN_ITS_REG_INV_OOR An attempt was made to invalidate an OOR interrupt.	0xE	Core, bits[31:16] Data, bits[15:0]	Software must ensure that the ID that is provided to GICR_INVLPIR is an LPI or vLPI ID. Also, GICR_WAKER.Sleep is not set and for physical LPIs that GICR_CTLR.Enable_LPI is set.

4.17.4.2 SPI RAM error records 1-2

SPI RAM error record 1 records RAM ECC errors that are correctable. SPI RAM error record 2 records RAM ECC errors that are uncorrectable.

The GIC-720AE has two SPI RAM, SPI0 and SPI1 that contain the programming for SPIs. SPI0 contains SPIs that have even-numbered IDs, and SPI1 contains SPIs that have odd-numbered IDs.

If a correctable error is detected in SPI RAM, it is corrected and the error is reported in error record 1. See 4.17.3 Error recovery and fault handling interrupts on page 98 for information about the error counters and interrupt generation options.

Correctable errors do not require software to take any action within the GIC. However, software can choose to track error locations in case a RAM row or column can be repaired, and the RAM has repair capability.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the SPI error records.

Table 4-8: SPI RAM errors, records 1-2

Record	GICT_ERR<n>MISC0.Data
1 = Correctable	• Bit location, bits[31:$\log_2$(SPIs)] • ID, bits[$\log_2$(SPIs) – 1:0] Where SPIs is the number of SPIs that the configuration supports.
2 = Uncorrectable	ID, bits[$\log_2$(SPIs) – 1:0]

For example, if a GIC configuration supports 512 SPIs then:

- GICT_ERR<n>MISC0.Data[31:9] is the bit location.
- GICT_ERR<n>MISC0.Data[8:0] is the ID.

To calculate the INTID:

- If ID ≤ 960, then INTID = 32 + ID.
- If ID > 960, then INTID = 4096 + ID.

The RAM address can be determined from the ID >> 1. ID[0] specifies the SPI RAM number.

Related information

SPI error recovery procedure on page 80

4.17.4.3 SGI RAM error records 3-4

SGI RAM error record 3 records RAM ECC errors that are correctable. SGI RAM error record 4 records RAM ECC errors that are uncorrectable.

The Distributor records a subset of the SGI programming, and stores this information in the SGI RAM, to ensure that it can make the correct routing decisions for SGIs.

If a correctable error is detected in SGI RAM, the error is corrected and the error is reported in error record 3. See 4.17.3 Error recovery and fault handling interrupts on page 98 for information about the error counters and interrupt generation options.

Correctable errors do not require software to take any action within the GIC. However, the GIC can choose to track error locations in case a RAM row or column can be repaired, and the RAM has repair capability.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the SGI error records.

Table 4-9: SGI RAM errors, records 3-4

Record	GICT_ERR<n>MISC0.Data
3 = Correctable	• Bit location, bits[(ceiling(cores / 16) × 16)]+ • Address, bits[(ceiling(cores / 16) × 16) − 1:0]
4 = Uncorrectable	Address, bits[(ceiling(cores / 16) × 16) − 1:0]

The RAM stores information for the same SGI for up to 16 cores on a single row. The corrupted SGI number is given by:

- address MOD 16 on cores (address − (address MOD 16)) to (address − (address MOD 16)) + 15

GICR_SGIDR contains default values for GICR_IGROUPR0, GICR_IGRPMODR0, and GICR_NSACR for each SGI.

When an SGI is in error, the GIC operates using the values that GICR_SGIDR contains.

Related information

SGI error recovery procedure on page 72

4.17.4.4 TGT_SPI RAM error records 5-6

TGT_SPI RAM error record 5, records RAM ECC errors that are correctable. TGT_SPI RAM error record 6, records RAM ECC errors that are uncorrectable. Each error generates an SPI interrupt.

The TGT_SPI RAM stores the top three pending SPIs or doorbells for each PE.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the TGT_SPI error records.

Table 4-10: TGT_SPI RAM errors, records 5-6

Record	GICT_ERR<n>MISC0.Data
5 = Correctable	• Bit location, bits[31:$\log_2$(cores)] • Address, bits[$\log_2$(cores) − 1:0]
6 = Uncorrectable	Address, bits[$\log_2$(cores) − 1:0]

The GIC can recover most uncorrectable errors that occur in the TGT_SPI RAM. However, if an SPI is activated while handling an error, then the GIC might not mask the interrupt so a spurious interrupt can occur.

The GIC automatically recovers any lost doorbells that might occur.

4.17.4.5 PPI RAM error records 7-8

PPI RAM error record 7 records RAM ECC errors that are correctable. PPI RAM error record 8 records RAM ECC errors that are uncorrectable.

Error records 7-8 record the errors from PPI RAM that contain GICR_IPRIORITYRn and GICR_IPRIORITYRnE information for PPIs and SGIs. PPI RAM also contains a buffer that stores generated SGIs when backpressure occurs.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the PPI error records.

Table 4-11: PPI RAM errors, records 7-8

Record	GICT_ERR<n>MISC0.Data
7 = Correctable	• PPI block, bits[19+] • Bit location, bits[18:12] • Offset, bits[11:8] • SGI/Int, bit[7] • Core, bits[6:0]
8 = Uncorrectable	• PPI block, bits[12+] • Offset, bits[11:8] • SGI/Int, bit[7] • Core, bits[6:0]

For uncorrectable errors, if:

Bit[7], SGI/Int == 0

Software must perform the recovery sequence that 4.10.5 PPI error recovery procedure on page 74 describes.

Bit[7], SGI/Int == 1

The GIC did not generate the SGI because an error occurred during SGI generation. Although an SGI generation error occurs, the GIC continues to operate normally.

4.17.4.6 LPI RAM error records 9-10

LPI RAM error record 9 records RAM ECC errors that are correctable. LPI RAM error record 10 records RAM ECC errors that are uncorrectable. Each error generates an LPI interrupt.

LPI RAM error records 9-10 are present if LPI support is configured.

The LPI RAM is the main LPI cache and it stores the LPI properties and pending information.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the LPI error records.

Table 4-12: LPI RAM errors, records 9-10

Record	GICT_ERR<n>MISC0.Data
9 = Correctable	• Bit location, bits[14+] • Pending, bits[13:12]. These bits indicate if there were pending interrupts in the cache at the time of the corruption. • LPI channel, bits[11:10] • Address, bits[9:0]
10 = Uncorrectable	• Pending, bits[13:12] • LPI channel, bits[11:10] • Address, bits[9:0]

When an uncorrectable error occurs, the data shown in the table is stored and the GICT_ERR<n>MISC1 register is updated to contain the RAM contents of the corrupted line. The line in RAM is dropped, and any pending interrupts that it might contain are lost.

For uncorrectable errors, software must perform the recovery sequence that 4.13.6 LPI error recovery procedure on page 89 describes.

4.17.4.7 PTS RAM error records 11-12

Pending Table System (PTS) RAM error record 11 records RAM ECC errors that are correctable. PTS RAM error record 12 records RAM ECC errors that are uncorrectable. Each error generates an LPI interrupt.

PTS RAM error records 11-12 are present if LPI support is configured.

Error records 11-12, record errors from the Pending table map cache.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the PTS error records.

Table 4-13: PTS RAM errors, records 11-12

Record	GICT_ERR<n>MISC0.Data
11 = Correctable	• Bit location, bits[31:4] • Address[3:0]
12 = Uncorrectable	Address[3:0]

No recovery is required for uncorrectable errors. The GIC continues to operate with a small but temporary performance hit.

4.17.4.8 TGT_LPI RAM error records 13-14

TGT_LPI RAM error record 13, records RAM ECC errors that are correctable. TGT_LPI RAM error record 14, records RAM ECC errors that are uncorrectable. Each error generates an LPI interrupt.

TGT_LPI RAM error records 13-14 are present if LPI support is configured.

Error records 13-14, record errors from the main TGT_LPI cache.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the TGT_LPI error records.

Table 4-14: TGT_LPI RAM errors, records 13-14

Record	GICT_ERR<n>MISC0.Data
13 = Correctable	• Bit location, bits[31:$\log_2$(cores)] • Address, bits[$\log_2$(cores) – 1:0]
14 = Uncorrectable	Address, bits[$\log_2$(cores) – 1:0]

For TGT_LPI error recovery, see 4.13.6 LPI error recovery procedure on page 89.

4.17.4.9 VICM RAM error records 15-16

Virtual ITS Communication Module (VICM) RAM error record 15 records RAM ECC errors that are correctable. VICM RAM error record 16 records RAM ECC errors that are uncorrectable. Each error generates a vICM interrupt.

VICM RAM error records 15-16 are present if GIC-720AE is configured to support GICv4.1.

Error records 15-16, record errors from the VICM RAM, which caches the vPE Configuration table.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the VICM error records. `vpe_width` is a configuration parameter that sets the number of vPEs that the GIC supports, that is, $2^{\texttt{vpe\_width}}$ vPEs.

Table 4-15: VICM RAM errors, records 15-16

Record	GICT_ERR<n>MISC0.Data
15 = Correctable	• Bit location, bits[31:$\log_2$`vpe_width`] • Address, bits[$\log_2$`vpe_width` – 1:0]
16 = Uncorrectable	Address, bits[$\log_2$`vpe_width` – 1:0]

4.17.4.10 VSPA RAM error records 17-18

Virtual SGI Pending Array (VSPA) RAM error record 17 records RAM ECC errors that are correctable. VSPA RAM error record 18 records RAM ECC errors that are uncorrectable. Each error generates a VSPA interrupt.

VSPA RAM error records 17-18 are present if GIC-720AE is configured to support GICv4.1.

Error records 17-18, record errors from the vICM search RAM.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the VSPA error records. `vpe_width` is a configuration parameter that sets the number of vPEs that the GIC supports, that is, 2^{vpe_width} vPEs.

Table 4-16: VSPA RAM errors, records 17-18

Record	GICT_ERR<n>MISC0.Data
17 = Correctable	When `vpe_width` ≤ 8: • Bit location, bits[31:1] • Address, bit[0] When `vpe_width` > 8: • Bit location, bits[31:$\log_2 2^{vpe\_width}$ / 128] • Address, bits[$\log_2(2^{vpe\_width}$ / 128) – 1:0]
18 = Uncorrectable	When `vpe_width` ≤ 8: • Address, bit[0] When `vpe_width` > 8: • Address, bits[$\log_2(2^{vpe\_width}$ / 128) – 1:0]

4.17.4.11 VTGT_VSTR RAM error records 19-20

Virtual Target Store (VTGT_VSTR) RAM error record 19 records RAM ECC errors that are correctable. VTGT_VSTR RAM error record 20 records RAM ECC errors that are uncorrectable. Each error generates a VTGT_VSTR interrupt.

VTGT_VSTR RAM error records 19-20 are present if GIC-720AE is configured to support GICv4.1.

Error records 19-20, record errors from the vTGT Store that stores the highest priority LPIs, vSGI, and doorbell information for each vPE.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the VTGT_VSTR error records. `vpe_width` is a configuration parameter that sets the number of vPEs that the GIC supports, that is, 2^{vpe_width} vPEs.

Table 4-17: VTGT_VSTR RAM errors, records 19-20

Record	GICT_ERR<n>MISC0.Data
19 = Correctable	• Bit location, bits[31:`vpe_width`] • Address, bits[`vpe_width` – 1:0]
20 = Uncorrectable	Address, bits[`vpe_width` – 1:0]

4.17.4.12 VTGT_VRES RAM error records 21-22

Virtual Target Residency (VTGT_VRES) RAM error record 21 records RAM ECC errors that are correctable. VTGT_VRES RAM error record 22 records RAM ECC errors that are uncorrectable. Each error generates a VTGT_VRES interrupt.

VTGT_VRES RAM error records 21-22 are present if GIC-720AE is configured to support GICv4.1.

Error records 21-22, record errors from the vTGT Residency RAM that stores the highest priority vLPIs and vSGI information for resident vPEs.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the VTGT_VRES error records.

Table 4-18: VTGT_VRES RAM errors, records 21-22

Record	GICT_ERR<n>MISC0.Data
21 = Correctable	• Bit location, bits[31:$\log_2$(cores)] • Address, bits[$\log_2$(cores) – 1:0]
22 = Uncorrectable	Address, bits[$\log_2$(cores) – 1:0]

4.17.4.13 VTGT_SRCH RAM error records 23-24

Virtual Target Search (VTGT_SRCH) RAM error record 23 records RAM ECC errors that are correctable. VTGT_SRCH RAM error record 24 records RAM ECC errors that are uncorrectable. Each error generates a VTGT_SRCH interrupt.

VTGT_SRCH RAM error records 23-24 are present if GIC-720AE is configured to support GICv4.1.

Error records 23-24, record errors from the VTGT_SRCH RAM, which the GIC uses for efficient searching of all vPEs.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the VTGT_SRCH error records. `vpe_width` is a configuration parameter that sets the number of vPEs that the GIC supports, that is, $2^{\texttt{vpe\_width}}$ vPEs.

Table 4-19: VTGT_SRCH RAM errors, records 23-24

Record	GICT_ERR<n>MISC0.Data
23 = Correctable	When `vpe_width` ≤ 8: • Bit location, bits[31:1] • Address, bit[0] When `vpe_width` > 8: • Bit location, bits[31:$\log_2 2^{\texttt{vpe\_width}}$ / 128] • Address, bits[$\log_2(2^{\texttt{vpe\_width}}$ / 128) – 1:0]
24 = Uncorrectable	When `vpe_width` ≤ 8: • Address, bit[0] When `vpe_width` > 8: • Address, bits[$\log_2(2^{\texttt{vpe\_width}}$ / 128) – 1:0]

4.17.4.14 ITS RAM error records 25-26

ITS RAM error record 25 records ITS RAM ECC errors that are correctable. ITS RAM error record 26 records ITS RAM ECC errors that are uncorrectable.

ITS RAM error records 25-26 are present if an ITS is configured.

Error records 25-26 record the errors from ITS RAM.

All ITS tables are memory backed allowing uncorrectable errors to be read from RAM again without software intervention. These records are used for tracking RAM errors and for possible RAM maintenance.

The following table shows the information that GICT_ERR<n>MISC0.Data provides for the ITS RAM error records.

Table 4-20: ITS RAM errors, records 25-26

Record	GICT_ERR<n>MISC0.Data
25 = Correctable	• Address, bits[31:x + 10] • Bit location, bits[x + 9:x + 2] • RAM, bits[x + 1:x] • ITS, bits[x – 1:0] Where x = $\log_2$(ITS)
26 = Uncorrectable	• Address, bits[31:x + 3)] • RAM, bits[x + 2:x] • ITS, bits[x – 1:0] Where x = $\log_2$(ITS)

GICT_ERR<n>MISC0 gives information relating to the corrupted ITS, RAM, and RAM address. The bit location of a correctable error is also given. The ITS RAM encoding is shown in the following table.

Table 4-21: ITS RAM encoding

RAM	Record 25	Record 26
0	None	None
1	Device cache	Device cache
2	Collection cache	Collection cache
3	Event cache	Event cache

4.17.4.15 ITS command and translation error records 27+

The ITS command and translation error records 27+ record uncorrectable command and translation errors from each configured ITS.

The ITS command and translation error records capture software events so that the operation of software can be tracked. The software command errors that are captured are uncorrectable errors only, which require software to correct the command to restart.

The GICT_ERR<n>STATUS.IERR field indicates whether an error is either related to the architecture (0) or implementation defined (1). In both cases, the full 24-bit syndrome is reported in GICT_ERR<n>MISC0. Extra data is reported in GICT_ERR<n>MISC1.

The data that is captured for each ITS software syndrome is shown in the following table.

Table 4-22: ITS command and translation errors, records 27+

`MAPD` commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MAPD_DEVICE_OOR	`0x10801`	0	1	CEE	A `MAPD` command has tried to map a device with a DeviceID that is outside the supported range, or that is beyond the memory allocated.	0
MAPD_ITTSIZE_OOR	`0x10802`	0	1	CEE	A command has tried to allocate an ITT table that is larger than the supported EventID size.	0

`MAPC` commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MAPC_COLLECTION_OOR	`0x10903`	0	1	CEE	A `MAPC` command has tried to map a CollectionID that is not supported. See GITS_TYPER.	-
MAPC_TGT_OOR	`0x10920`	1	1	CEE	A `MAPC` command has tried to map to a core that does not exist.	-
MAPC_SRC_CHIP_OOR	`0x10922`	1	0	-	Specified targetPE (RDnum) references an out-of-range, nonexistent chip.	RDbase from command
MAPC_SRC_TGT_OFF	`0x10923`	1	0	-	Specified targetPE (RDnum) has GICR_CTLR.EnableLPI = 0.	RDbase from command

MAPC commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MAPC_SRC_CHIP_OFF	0x10925	1	0	-	Specified targetPE (RDnum) references an offline chip. See GICD_CHIPRn.	RDbase from command

MAPI commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MAPI_DEVICE_OOR	0x10B01	0	1	CEE	A MAPI has tried to map a DeviceID that is not supported. See GITS_BASER0, and for information about the supported range, see GITS_TYPER.	0
MAPI_COLLECTION_OOR	0x10B03	0	1	CEE	A MAPI has tried to map to a collection that is not supported. See GITS_BASER1, and for information about the supported range, see GITS_TYPER.	0
MAPI_UNMAPPED_DEVICE	0x10B04	0	1	CEE	A MAPI has tried to map an interrupt to a device that is not mapped.	0
MAPI_ID_OOR	0x10B05	0	1	CEE	A MAPI has tried to map to an EventID size that is not supported. The size that is supported is reported in GITS_TYPER, but might be reduced depending on the MAPD command for the specified DeviceID.	0

MAPTI commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MAPTI_DEVICE_OOR	0x10A01	0	1	-	Specified DeviceID is outside of configured or allocated range.	0
MAPTI_COLLECTION_OOR	0x10A03	0	1	-	Specified CollectionID is outside of the configured or allocated range.	0
MAPTI_UNMAPPED_DEVICE	0x10A04	0	1	-	Specified DeviceID has not been allocated with previous MAPD command.	0
MAPTI_ID_OOR	0x10A05	0	1	-	Specified EventID is outside the range allocated with ITTSize on the relevant MAPD command.	0
MAPTI_PHYSICALID_OOR	0x10A06	0	1	-	Specified physical INTID is greater than 16 bits. If the Redistributor allocates a smaller PID range, then this is reported on incoming LPI and other relevant ITS commands that reach the Redistributor.	0

MOVI commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MOVI_DEVICE_OOR	0x10101	0	1	CEE	A MOVI has tried to map a device that is outside the range that the ITS supports. See GITS_BASER0, and for information about the supported range, see GITS_TYPER.	0
MOVI_COLLECTION_OOR	0x10103	0	1	CEE	A MOVI has tried to use a collection that is outside the range that the ITS supports. See GITS_BASER1, and for information about the supported range, see GITS_TYPER.	0
MOVI_UNMAPPED_DEVICE	0x10104	0	1	CEE	A MOVI has tried to move an interrupt from a device that is not mapped.	0

MOVI commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MOVI_ID_OOR	0x10105	0	1	CEE	A MOVI has tried to use an EventID that is outside the size that the corresponding MAPD command supports.	0
MOVI_UNMAPPED_INTERRUPT	0x10107	0	1	CEE	A MOVI command has tried to operate on an interrupt that is not mapped.	0
MOVI_ID_IS_VIRTUAL	0x10108	0	1	-	Specified DeviceID/EventID pair has been mapped as a virtual LPI and so a VMOVI command must be used.	0
MOVI_UNMAPPED_COLLECTION	0x10109	0	1	CEE	A MOVI command has tried to operate on a collection that is not mapped.	0
MOVI_SRC_TGT_OOR	0x10120	1	0	-	Specified DeviceID/EventID pair has been mapped to a nonexistent target on an online chip by previous commands.	RD that LPI is mapped to
MOVI_DST_TGT_OOR	0x10121	1	0	-	Specified target collection (ICID) is mapped to a nonexistent target on an online chip by previous commands.	RD that specified collection ICID is mapped to
MOVI_SRC_CHIP_OOR	0x10122	1	0	-	Specified DeviceID/EventID pair has been mapped to an out-of-range chip by previous commands.	RD that LPI is mapped to
MOVI_SRC_TGT_OFF	0x10123	1	0	-	Specified DeviceID/EventID pair is mapped to a PE with GICR_CTLR.EnableLPI = 0.	RD that LPI is mapped to
MOVI_DST_TGT_OFF	0x10124	1	0	-	Specified target collection (ICID) is mapped to a PE with GICR_CTLR.EnableLPI = 0.	RD that specified collection ICID is mapped to
MOVI_SRC_CHIP_OFF	0x10125	1	0	-	Specified DeviceID/EventID pair is mapped to an offline chip. See GICD_CHIPRn.	RD that LPI is mapped to
MOVI_DST_CHIP_OOR	0x10128	1	0	-	Specified target collection (ICID) is mapped to an out-of-range or nonexistent chip.	RD that specified collection ICID is mapped to
MOVI_DST_CHIP_OFF	0x10129	1	0	-	Specified target collection (ICID) is mapped to an offline chip. See GICD_CHIPRn.	RD that specified collection ICID is mapped to

MOVALL commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MOVALL_SRC_TGT_OOR	0x10E20	1	0	-	Specified RDbase1 references a nonexistent target on an online chip. If MISC1 data is 0, then either RDbase1 or RDbase2 are greater than the hardware supports.	RDbase1 from command or 0
MOVALL_DST_TGT_OOR	0x10E21	1	0	-	MOVALL to a core that does not exist. Command is ignored.	RDbase2 from command
MOVALL_SRC_CHIP_OOR	0x10E22	1	0	-	Specified RDbase1 is on an out-of-range nonexistent chip.	RDbase1 from command

MOVALL commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
MOVALL_SRC_TGT_OFF	0x10E23	1	0	-	Specified RDbase1 has GICR_CTLR.EnableLPI = 0.	RDbase1 from command
MOVALL_DST_TGT_OFF	0x10E24	1	0	-	Specified RDbase2 has GICR_CTLR.EnableLPI = 0.	RDbase2 from command
MOVALL_SRC_CHIP_OFF	0x10E25	1	0	-	Specified RDbase1 is on an offline chip. See GICD_CHIPRn.	RDbase1 from command
MOVALL_DST_CHIP_OOR	0x10E28	1	0	-	Specified RDbase2 is on an out-of-range nonexistent chip.	RDbase2 from command
MOVALL_DST_CHIP_OFF	0x10E29	1	0	-	Specified RDbase2 is on an offline chip. See GICD_CHIPRn.	RDbase2 from command

DISCARD commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
DISCARD_DEVICE_OOR	0x10F01	0	1	CEE	A DISCARD has tried to use a device that is outside the range that the ITS supports. See GITS_BASER0, and for information about the supported range, see GITS_TYPER.	0
DISCARD_UNMAPPED_DEVICE	0x10F04	0	1	CEE	A DISCARD has tried to drop an interrupt from a device that is not mapped.	0
DISCARD_ID_OOR	0x10F05	0	1	CEE	A DISCARD command has tried to use an EventID that is outside the size that the corresponding MAPD command supports.	0
DISCARD_UNMAPPED_INTERRUPT	0x10F07	0	1	CEE	A MOVI command has tried to operate on an interrupt that is not mapped.	0
DISCARD_ITE_INVALID	0x10F10	0	1	CEE	A MOVI command has tried to operate on an EventID that the corresponding MAPD command does not support.	0

CLEAR commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
CLEAR_DEVICE_OOR	0x10501	0	1	CEE	A CLEAR has attempted to use a device that is outside the range that the ITS supports. See GITS_BASER0, and for information about the supported range, see GITS_TYPER.	0
CLEAR_UNMAPPED_DEVICE	0x10504	0	1	CEE	A CLEAR has tried to drop an interrupt from a device that is not mapped.	0
CLEAR_ID_OOR	0x10505	0	1	CEE	A CLEAR has tried to drop an interrupt from an EventID that the corresponding MAPD command does not support.	0

CLEAR commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
CLEAR_UNMAPPED_INTERRUPT	0x10507	0	1	CEE	A CLEAR has attempted to drop an interrupt that is not mapped.	0
CLEAR_ITE_INVALID	0x10510	0	1	CEE	A CLEAR has tried to drop an interrupt from an EventID that the corresponding MAPD command does not support.	0
CLEAR_SRC_TGT_OOR	0x10520	1	0	-	Specified DeviceID/EventID pair has been mapped to a nonexistent target on an online chip by previous commands.	RD that LPI is mapped to
CLEAR_SRC_CHIP_OOR	0x10522	1	0	-	Specified DeviceID/EventID pair has been mapped to an out-of-range chip by previous commands.	RD that LPI is mapped to
CLEAR_SRC_TGT_OFF	0x10523	1	0	-	Specified DeviceID/EventID pair is mapped to a PE with GICR_CTLR.EnableLPI = 0.	RD that LPI is mapped to
CLEAR_SRC_CHIP_OFF	0x10525	1	0	-	Specified DeviceID/EventID pair is mapped to an offline chip. See GICD_CHIPRn.	RD that LPI is mapped to
CLEAR_PHYSICAL_ID_OOR	0x10526	1	0	-	A CLEAR has tried to drop an interrupt, which has a physical ID that the target does not support.	pIntID of LPI
VCLEAR_VID_OOR	0x12526	1	0	-	Specified DeviceID/EventID pair is mapped to a vIntID that is outside the specified vPT size range for its vPEID.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VCLEAR_NO_MAP	0x12530	1	0	-	Sent VCLEAR command to a vPEID that is not mapped on its ITS.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VCLEAR_VPE_OOR	0x12531	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VCLEAR_CHIP_OFF	0x12525	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that is currently mapped to an offline chip by previous VMAPP or VMOVP ID.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VCLEAR_VID_OOR_CC	0x12532	1	0	-	Specified DeviceID/EventID pair is mapped to a vIntID that is outside the specified vPT size range for its vPEID and the error is detected on a remote chip.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VCLEAR_VPE_LOST	0x12533	1	0	-	Specified DeviceID/EventID pair is mapped to a vPE that the system has lost. The causes for this issue can be taking chips offline, data corruption, or conflicting programming such as illegal VMAPP sequences.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}

INV commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
INV_DEVICE_OOR	0x10C01	0	1	CEE	An INV has tried to use a device that is outside the range that the ITS supports. See GITS_BASER0, and for information about the supported range, see GITS_TYPER.	0
INV_UNMAPPED_DEVICE	0x10C04	0	1	CEE	An INV has tried to invalidate an interrupt from a device that is not mapped.	0
INV_ID_OOR	0x10C05	0	1	CEE	An INV has tried to use an EventID that is outside the size that the corresponding MAPD command supports.	0
INV_UNMAPPED_INTERRUPT	0x10C07	0	1	CEE	An INV has tried to invalidate an interrupt that is not mapped.	0
INV_ITE_INVALID	0x10C10	0	1	CEE	An INV has tried to invalidate an interrupt with an EventID that is invalid.	0
INV_SRC_TGT_OOR	0x10C20	1	0	-	Specified DeviceID/EventID pair has been mapped to a nonexistent target on an online chip by previous commands.	RD that LPI is mapped to
INV_SRC_CHIP_OOR	0x10C22	1	0	-	Specified DeviceID/EventID pair has been mapped to an out-of-range chip by previous commands.	RD that LPI is mapped to
INV_SRC_TGT_OFF	0x10C23	1	0	-	Specified DeviceID/EventID pair is mapped to a PE with GICR_CTLR.EnableLPI = 0.	RD that LPI is mapped to
INV_SRC_CHIP_OFF	0x10C25	1	0	-	Specified DeviceID/EventID pair is mapped to an offline chip. See GICD_CHIPRn.	RD that LPI is mapped to
INV_PHYSICAL_ID_OOR	0x10C26	1	0	-	An INV has tried to invalidate an interrupt with a physical ID that is larger than the target supports. See GICR_PROPBASER.IDbits.	pIntID of LPI
VINV_VID_OOR	0x12C26	1	0	-	Specified DeviceID/EventID pair is mapped to a vINTID that is outside the specified vPT size range for its vPEID.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VINV_NO_MAP	0x12C30	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that is not mapped on its ITS.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VINV_VPE_OOR	0x12C31	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VINV_CHIP_OFF	0x12C25	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that is currently mapped to an offline chip by previous VMAPP or VMOVP ID.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VINV_VID_OOR_CC	0x12C32	1	0	-	Specified DeviceID/EventID pair is mapped to a vINTID that is outside the specified vPT size range for its vPEID and it is detected on a remote chip.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}

INV commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
VINV_VPE_LOST	0x12C33	1	0	-	Specified DeviceID/EventID pair is mapped to a vPE that the system has lost. The causes of this issue can be taking chips offline, data corruption, or conflicting programming such as illegal VMAPP sequences.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}

INVALL commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
INVALL_COLLECTION_OOR	0x10D03	0	1	CEE	An INVALL has tried to invalidate an OOR collection. See GITS_TYPER.	0
INVALL_UNMAPPED_COLLECTION	0x10D09	0	1	CEE	An INVALL has tried to invalidate a collection that is not mapped.	0
INVALL_SRC_TGT_OOR	0x10D20	1	0	-	An INVALL has been sent to an illegal target.	RD that collection ICID is mapped to
INVALL_SRC_CHIP_OOR	0x10D22	1	0	-	An INVALL has tried to invalidate an interrupt from a device that is not mapped.	RD that collection ICID is mapped to
INVALL_SRC_TGT_OFF	0x10D23	1	0	-	An INVALL has been sent to a target that has LPIs turned off.	RD that collection ICID is mapped to
INVALL_SRC_CHIP_OFF	0x10D25	1	0	-	Specified collection (ICID) is mapped to an offline chip. See GICD_CHIPRn.	RD that collection ICID is mapped to
VINVALL_VCPU_OOR	0x12D03	0	1	-	Specified vPEID that is outside the hardware maximum or GITS_BASER2 configured range.	0
VINVALL_NO_MAP	0x12D30	1	0	-	Specified vPEID that is not mapped on the ITS.	vPEID
VINVALL_VPE_OOR	0x12D31	1	0	-	Specified vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	vPEID
VINVALL_CHIP_OFF	0x12D25	1	0	-	Specified vPEID is currently mapped to an offline chip by previous VMAPP or VMOVP commands.	{chip[CHIP_ID_WIDTH-1:0], 0x0000, vPEID[VPE_WIDTH-1:0]}
VINVALL_VPE_LOST	0x12D33	1	0	-	The system has lost the specified vPE. The causes of this issue can be taking chips offline, data corruption, or conflicting programming such as illegal VMAPP sequences.	vPEID

INT commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**Data (data is only for GITS_TRANSLATER write interrupts not for INT commands)**
INT_DEVICE_OOR	0x10301	0	0, 1	UEE	An incoming translation has attempted to use a device that is outside the range that the ITS supports. See GITS_BASER0, and for information about the supported range, see GITS_TYPER.	If not stalled: [31:0] DID
INT_UNMAPPED_DEVICE	0x10304	0	0, 1	UEE	An incoming translation has tried to invalidate an interrupt from a device that is not mapped.	If not stalled: [23:0] DID
INT_ID_OOR	0x10305	0	0, 1	UEE	An INT has tried to use an EventID that is outside the size that the corresponding MAPD command supports. The debug data bit[50] is the OR reduction of VID bits[31:20] as indicated by |VID[31:20].	If not stalled: [50] VID[31:20] contains 1. [43:24] VID[19:0] [23:0] DID
INT_UNMAPPED_INTERRUPT	0x10307	0	0, 1	UEE	An INT command has tried to raise an interrupt that is not mapped. The debug data bit[50] is the OR reduction of VID bits[31:20] as indicated by |VID[31:20].	If not stalled: [50] VID[31:20] contains 1. [43:24] VID [23:0] DID
INT_ITE_INVALID	0x10310	0	0, 1	UEE	An INT command has tried to raise an interrupt with an EventID that the corresponding MAPD command does not support.	If not stalled: [13:0] Collection ID
INT_TGT_OFF	0x10323	1	0	-	INT received for a target with GICR_CTLR.Enable_LPIs disabled. Software must either enable LPI or correct mappings. Target is reported in GICT_ERR<n>MISC1.	RD that LPI is mapped to
INT_CHIP_OFF	0x10325	1	0	-	Specified DeviceID/EventID pair is mapped to a PE of an offline chip. See GICD_CHIPRn.	RD that LPI is mapped to
INT_PHYSICALID_OOR	0x10326	1	0	-	INT received with a physical ID that is beyond the range that is specified in GICR_PROPBASER.IDbits. Software must correct mappings. Interrupt is dropped and ID is reported in GICT_ERR<n>MISC1.	RD that LPI is mapped to
VLPI_VID_OOR	0x12426	1	0	-	Specified DeviceID/EventID pair is mapped to a vINTID that is outside the specified vPT size range for its vPEID.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}

INT commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	Data (data is only for GITS_TRANSLATER write interrupts not for INT commands)
VLPI_NO_MAP	0x12430	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that is not mapped on its ITS.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VLPI_VPE_OOR	0x12431	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VLPI_CHIP_OFF	0x12425	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that is currently mapped to an offline chip by previous VMAPP or VMOVP commands.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VLPI_VID_OOR_CC	0x12432	1	0	-	Specified DeviceID/EventID pair is mapped to a vINTID that is outside the specified vPT size range for its vPEID and the error is detected on a remote chip.	{chip[CHIP_ID_WIDTH-1:0], vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VLPI_VPE_LOST	0x12433	1	0	-	Specified DeviceID/EventID pair is mapped to a vPEID that the system has lost. The causes of this issue can be taking chips offline, data corruption, or conflicting programming such as illegal VMAPP sequences.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}

VMAPP commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
VMAPP_VCPU_OOR	0x12903	0	1	-	Specified vPEID that is outside the hardware maximum or GITS_BASER2 configured range.	0
VMAPP_PHYSICALID_OOR	0x12904	0	1	-	Specified PID (doorbell ID) is above the hardware maximum range or below 8192 except 1023.	0
VMAPP_VPTSIZE_OOR	0x12910	0	1	-	Specified VPTsize outside of hardware maximum range.	0
VMAPP_TGT_FULL_OOR	0x12920	1	0	-	Specified Target (RDnum) is outside of hardware range.	0
VMAPP_TGT_OOR	0x12921	1	0	-	Specified Target (RDnum) does not exist but does reference an online chip.	RDbase from command
VMAPP_ENLPI_OFF	0x12924	1	0	-	Specified Target (RDnum) has GICR_CTLR.EnableLPI = 0.	{chip[CHIP_ID_WIDTH-1:0], DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_CHIP_OFF	0x12925	1	0	-	Specified Target (RDnum) is on an offline chip. See GICD_CHIPRn.	{chip[CHIP_ID_WIDTH-1:0], DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}

VMAPP commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
VMAPP_DBID_OOR	0x12926	1	0	-	Specified PID (doorbell ID) is outside the range supported by GICR_PROPBASER programming.	Ignore data.
VMAPP_CHIP_OOR	0x12928	1	0	-	Specified Target (RDnum) is on a nonexistent chip.	{chip[CHIP_ID_WIDTH-1:0], DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_DST_CHIP_OFF	0x12929	1	0	-	Specified Target (RDnum) is an offline chip. See GICD_CHIPRn.	{chip[CHIP_ID_WIDTH-1:0], DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_NO_MAP	0x12930	1	0	-	Specified vPEID when V=0 has not been previously mapped.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_VPE_OOR	0x12931	1	0	-	Specified vPEID is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_VPE_LOST	0x12933	1	0	-	The system has lost the specified vPEID. The causes of this issue can be taking chips offline, data corruption, or conflicting programming such as illegal VMAPP sequences.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_ACE_LITE_VPT_RD_FAILURE	0x12934	1	0	-	vPT read access performed as a part of a VMAPP command received SLVERR or DECODE error.	{chip[CHIP_ID_WIDTH-1:0], DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_VPROP_V	0x12936	1	0	-	Specified Target (RDnum) does not have GICR_VPROPBASER.Valid set.	{chip[CHIP_ID_WIDTH-1:0], DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_VPE_OOR_CC	0x12938	1	0	-	Specified vPEID is outside the GITS_BASER2 and GICR_VPROPBASER configured range and is detected on the remote chip due illegal different ranges on different chips.	{chip[CHIP_ID_WIDTH-1:0], DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMAPP_VPE_CFG_TOP_INV	0x12939	1	0	-	Specified vPEID maps to an invalid L1 entry in indirect vPE config table.	{chip[CHIP_ID_WIDTH-1:0], DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}

VMAPI commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
VMAPI_DEVICE_OOR	0x12B01	0	1	-	Specified DeviceID outside of the hardware maximum or GITS_BASER0 configured range.	0
VMAPI_VCPU_OOR	0x12B03	0	1	-	Specified vPEID that is outside the hardware maximum or GITS_BASER2 configured range.	0
VMAPI_UNMAPPED_DEVICE	0x12B04	0	1	-	Specified DeviceID has not been allocated with previous MAPD command.	0
VMAPI_ID_OOR	0x12B05	0	1	-	Specified EventID is outside the range allocated with ITTSize on the relevant MAPD command.	0

VMAPTI commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
VMAPTI_DEVICE_OOR	0x12A01	0	1	-	Specified DeviceID outside of the hardware maximum or GITS_BASER0 configured range.	0
VMAPTI_VCPU_OOR	0x12A03	0	1	-	Specified vPEID that is outside the hardware maximum or GITS_BASER2 configured range.	0
VMAPTI_UNMAPPED_DEVICE	0x12A04	0	1	-	Specified DeviceID has not been allocated with previous MAPD command.	0
VMAPTI_ID_OOR	0x12A05	0	1	-	Specified EventID is outside the range allocated with ITTSize on the relevant MAPD command.	0
VMAPTI_VIRTUALID_OOR	0x12A13	0	1	-	Specified vID that above the hardware maximum range or below 8192.	0

VMOVP commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
VMOVP_VCPU_OOR	0x12203	0	1	-	Specified vPEID that is outside the hardware maximum or GITS_BASER2 configured range.	0
VMOVP_PHYSICALID_OOR	0x12204	0	1	-	Specified doorbell PID that above the hardware maximum range or below 8192 except 1023.	0
VMOVP_TGT_FULL_OOR	0x12220	1	0	-	Specified target (RDnum) that is outside of the hardware supported range.	0
VMOVP_TGT_OOR	0x12221	1	0	-	Specified Target (RDnum) does not exist but does reference an online chip.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_ENLPI_OFF	0x12224	1	0	-	Specified Target (RDnum) has GICR_CTLR.EnableLPI = 0.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_CHIP_OFF	0x12225	1	0	-	Specified vPEID is currently mapped to an offline chip by previous commands.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_DBID_OOR	0x12226	1	0	-	Specified doorbell PID is outside range supported by GICR_VPROPBASER programming.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_DST_CHIP_OOR	0x12228	1	0	-	Specified Target (RDnum) is on an out-of-range nonexistent chip.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_DST_CHIP_OFF	0x12229	1	0	-	Specified Target (RDnum) is on an offline chip. See GICD_CHIPRn.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_NO_MAP	0x12230	1	0	-	Specified vPEID has not been previously mapped on this ITS.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_VPE_OOR	0x12231	1	0	-	Specified vPEID is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}

VMOVP commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
VMOVP_VPE_LOST	0x12233	1	0	-	Specified vPEID is mapped to a vPEID that the system has lost. The causes of this issue can be taking chips offline, data corruption, or conflicting programming such as illegal VMAPP sequences.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_ACE_LITE_VPT_RD_FAILURE	0x12234	1	0	-	vPT read access performed as a part of a VMOVP command received SLVERR or DECODE error.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_VPROP_V	0x12236	1	0	-	Specified Target (RDnum) targets a CPU that does not have GICR_VPROPBASER.Valid set.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_VPE_REMAP	0x12237	1	0	-	Sent VMOVP command moving a vPE to a chip that already has this vPEID mapped. This issue can only occur if conflicting mapping are made for the same vPE by illegal software.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVP_VPE_OOR_CC	0x12238	1	0	-	Sent VMOVP command for a vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range on the destination chip. This issue can only occur if conflicting ranges are programmed on different chips by illegal software.	{DoorbellID[15:0], vPEID[VPE_WIDTH-1:0]}

VMOVI commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
VMOVI_DEVICE_OOR	0x12101	0	1	-	Specified DeviceID outside of the hardware maximum or GITS_BASER0 configured range.	0
VMOVI_VCPU_OOR	0x12103	0	1	-	Specified vPEID is outside the hardware maximum or GITS_BASER2 configured range.	0
VMOVI_UNMAPPED_DEVICE	0x12104	0	1	-	Specified DeviceID has not been allocated with previous MAPD command.	0
VMOVI_ID_OOR	0x12105	0	1	-	Specified EventID is outside the range allocated with ITTSize on the relevant MAPD command.	0
VMOVI_UNMAPPED_INTERRUPT	0x12107	0	1	-	Specified DeviceID/EventID pair has not been mapped by a previous MAPI or MAPTI command.	0
VMOVI_ID_IS_PHYSICAL	0x12115	0	1	-	Specified DeviceID/EventID has been mapped to physical LPI. MOVI command must be used.	0
VMOVI_VID_OOR	0x12126	1	0	-	Specified DeviceID/EventID mapped to a vINTID that is outside the specified vPT size range for its vPEID.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}

VMOVI commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
VMOVI_NO_MAP	0x12130	1	0	-	Specified vPE is not mapped on this ITS. This can happen if the vPE is corrupted by memory system errors or bad programming.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_VPE_OOR	0x12131	1	0	-	Specified DeviceID/EventID has been mapped to a vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_CHIP_OFF	0x12125	1	0	-	Specified DeviceID/EventID has been mapped to an offline chip.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_VID_OOR_CC	0x12132	1	0	-	Specified DeviceID/EventID has been mapped to a vINTID that is outside the specified vPT size range for its vPEID on a remote chip.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_VPE_LOST	0x12133	1	0	-	Specified DeviceID/EventID has been mapped to a vPEID that the system has lost. The causes of this issue can be taking chips offline, data corruption, or conflicting programming such as illegal VMAPP sequences.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_DST_NO_MAP	0x12140	1	0	-	Specified destination vPEID that is not mapped on ITS.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_DST_VPE_OOR	0x12141	1	0	-	Specified destination vPEID is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_DST_VID_OOR	0x12146	1	0	-	Specified DeviceID/EventID has been mapped to a vINTID that is outside the specified vPT size range for its destination vPEID.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_DST_CHIP_OFF	0x12129	1	0	-	Specified destination vPEID is currently mapped to an offline chip.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_DST_VID_OOR_CC	0x12142	1	0	-	Specified DeviceID/EventID has been mapped to a vINTID that is outside the specified vPT size range for its destination vPEID on a remote chip.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}
VMOVI_DST_VPE_LOST	0x12143	1	0	-	The system has lost the specified destination vPEID. The causes of this issue can be taking chips offline, data corruption, or conflicting programming such as illegal VMAPP sequences.	{vIntID[15:0], vPEID[VPE_WIDTH-1:0]}

INVDB commands						
Error mnemonic	Encoding	IERR	Stall	Mask	Description	MISC1 data
INVDB_VCPU_OOR	0x12E03	0	1	-	INVDB specified vPEID that is outside the hardware maximum or GITS_BASER2 configured range.	0
INVDB_NO_MAP	0x12E30	1	0	-	Sent INVDB command to a vPEID that is not mapped on its ITS.	vPEID

INVDB commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
INVDB_VPE_OOR	0x12E31	1	0	-	Sent INVDB command for a vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	vPEID
INVDB_CHIP_OFF	0x12E25	1	0	-	Sent INVDB command targeted to an offline chip.	{chip[CHIP_ID_WIDTH-1:0], 0x0000, vPEID[VPE_WIDTH-1:0]}
INVDB_VPE_LOST	0x12E33	1	0	-	Sent INVDB command for a vPEID that has inconsistent mappings in the system.	vPEID

VSGI commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
VSGI_VCPU_OOR	0x12303	0	1	-	VSGI command specified a vPEID that is outside the hardware maximum or GITS_BASER2 configured range.	0
VSGI_CMD_NO_MAP	0x12330	1	0	-	Sent VSGI command to a vPEID that is not mapped on its ITS.	{Priority[3:0], 0b0, Enable, Group, PendingClear, vIntID[3:0], vPEID[VPE_WIDTH-1:0]}
VSGI_CMD_VPE_OOR	0x12331	1	0	-	Sent VSGI command for a vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	{Priority[3:0], 0b0, Enable, Group, PendingClear, vIntID[3:0], vPEID[VPE_WIDTH-1:0]}
VSGI_CMD_CHIP_OFF	0x12325	1	0	-	Sent VSGI command targeted to an offline chip.	{chip[CHIP_ID_WIDTH-1:0], 0x0, Priority[3:0], 0b0, Enable, Group, PendingClear, vIntID[3:0], vPEID[VPE_WIDTH-1:0]}
VSGI_CMD_VPE_LOST	0x12333	1	0	-	Sent VSGI command for a vPEID that has inconsistent mappings in the system.	{Priority[3:0], 0b0, Enable, Group, PendingClear, vIntID[3:0], vPEID[VPE_WIDTH-1:0]}
VSGI_CMD_ACE_LITE_VPT_RD_FAILURE	0x12334	1	0	-	vPT read access performed as a part of a VSGI command received SLVERR or DECODE error.	{chip[CHIP_ID_WIDTH-1:0], 0x0, Priority[3:0], 0b0, Enable, Group, PendingClear, vIntID[3:0], vPEID[VPE_WIDTH-1:0]}

Implementation-defined features, non-virtual commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
OPR_DEVICE_OOR	0x100C0	1	-	CEE	Software has tried an operation through GITS_OPR using a device that is outside the range that the ITS supports. See GITS_BASER0, and for information about the supported range, see GITS_TYPER.	0

Implementation-defined features, non-virtual commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
OPR_UNMAPPED_COLLECTION	0x100C1	1	-	CEE	Software has tried an operation through GITS_OPR using a collection that is outside the range that the ITS supports. See GITS_BASER0, and for information about the supported range, see GITS_TYPER.	0
OPR_ID_OOR	0x100C2	1	-	CEE	Software has tried to lock an interrupt using an EventID that is larger than the specified device supports. The GITS_OPSR register reports a fail.	0
OPR_UNMAPPED_DEVICE	0x100C3	1	-	CEE	Software has tried to lock an interrupt from a device that is not mapped through GITS_OPR. The GITS_OPSR register reports a fail.	0
OPR_UNMAPPED_INTERRUPT	0x100C5	1	-	CEE	Software has tried to lock an interrupt that is not mapped through GITS_OPR. The GITS_OPSR register reports a fail.	0
OPR_SET_LOCKED	0x100C6	1	-	CEE	Software has tried to lock an interrupt into the cache but the set already contains a locked interrupt. The GITS_OPSR register reports a fail.	0
ACE_LITE_ACCESS_FAILURE_CMD	0x100C8	1	-	-	An access that the ITS issues, receives an SLVERR or DECODE error. The address is given in GICT_ERR<n>MISC1. This error can occur from multiple sources. Software must determine whether the Command queue is stalled, by checking GITS_CREADR.Stalled. If the Command queue has stalled, the command might not have occurred. See 4.12.3 ITS commands and errors on page 85.	[50:0] ACE-Lite manager address [51:1]
ACE_LITE_ACCESS_FAILURE_TRANSR	0x100C9	1	0	-	An access that the ITS issues for an interrupt, receives an SLVERR or DECODE error. The address is given in GICT_ERR<n>MISC1. This error can occur from multiple sources.	[50:0] ACE-Lite manager address [51:1]
ACE_LITE_ACCESS_FAILURE_LOCK	0x100CA	1	0	-	An access that the ITS issues for an OPR request, receives an SLVERR or DECODE error. The address is given in GICT_ERR<n>MISC1. This error can occur from multiple sources.	[50:0] ACE-Lite manager address [51:1]

Implementation-defined features, non-virtual commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
ACE_LITE_TRANS_FAILURE	0x100CB	1	-	AEE	An unknown source in the system has written to the subordinate port with an access that is not a legal GITS_TRANSLATER access. The full address of the access is given in GICT_ERR<n>MISC1. If the address matches GITS_TRANSLATER, then the size, length, strobes, or access type is wrong. Read accesses are not tracked.	[15:0] ACE-Lite subordinate address [15:0]
ACE_LITE_ADDR_OOR	0x100CC	1	-	-	ITS programming has tried to create an access to the address specified in GICT_ERR<n>MISC1 that is larger than the address space supported.	[50:0] ACE-Lite manager address [51:1]
INVALID_MULTI_LEVEL_DEV_TABLE_ENTRY	0x100CD	1	1/0	-	Software is using a two-level Device table and the first level table entry has not completed for command. Software must allocate and clear a new second-level table, update the first-level entry, and repeat the command.	0
INVALID_MULTI_LEVEL_DEV_TABLE_ENTRY_LOCK	0x100CE	1	0	-	Software is using a two-level Device table and the first level table entry has not completed for OPR request. Software must allocate and clear a new second-level table, update the first-level entry, and repeat the command.	0
IMDEF_INVALID_COMMAND	0x100CF	1	1	-	ITS command queue read an invalid opcode. When gicv41_support==1, this error can also indicate that a command requiring GICv4.1 command support has been detected but with GITS_BASER2.Valid==0.	0

Implementation-defined features, virtual commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
BASER2_DATA_ERR	0x12051	1	0	-	Writing GITS_BASER2.Valid with data mismatching with the existing vPE Configuration table.	0

4.17.4.16 CC RAM error records 62-63

Cross-chip (CC) RAM error record 62 records RAM ECC errors that are correctable. CC RAM error record 63 records RAM ECC errors that are uncorrectable.

The GICT_ERR<n>MISC0 register reports data for CC error records 62-63 shown in the following table.

Table 4-23: CC RAM errors, records 62-63

Record	GICT_ERR<n>MISC0.Data
62 = Correctable	• Bit location, bits[5+x:x] • Address, bits[x−1:0]
63 = Uncorrectable	Where x = $\log_2$(chip_count × ((gicv41_support & vsgi_cc_tokens) + (lpi_support × lpi_cc_tokens) + sgi_cc_tokens)).

4.17.4.17 Clearing error records

After reading a GICT_ERR<n>STATUS register, software must clear the valid register bits so that any new errors are recorded.

During this period, a new error might overwrite the syndrome for the error that was read previously. If the register is read or written, the previous error is lost.

To prevent this, most bits use a modified version of write-1-to-clear:

- Writes to the GICT_ERR<n>STATUS.UE (uncorrectable error records) or GICT_ERR<n>STATUS.CE (correctable error records) bits are ignored if GICT_ERR<n>STATUS.OF is set and is not being cleared.
- Writes to other fields in the GICT_ERR<n>STATUS register are ignored if either GICT_ERR<n>STATUS.UE or GICT_ERR<n>STATUS.CE are set and are not being cleared.

Similarly, GICT_ERR<n>MISC0 and GICT_ERR<n>MISC1 cannot be written, except the counter fields, if the corresponding GICT_ERR<n>STATUS.MV bit is set, and GICT_ERR<n>ADDR cannot be written if GICT_ERR<n>STATUS.AV is set.

Related information

SGI error recovery procedure on page 72
PPI error recovery procedure on page 74
SPI error recovery procedure on page 80
LPI error recovery procedure on page 89

4.17.5 Bus errors

ACE5-Lite bus error syndromes such as bad transactions, and corrupted RAM data reads can be made to report an ACE5-Lite external AXI *Subordinate Error* (SLVERR).

The GICT_ERR0CTLR.UE bit can be used to enable the SLVERR ACE5-Lite bus error for the syndromes shown in the following table.

Table 4-24: Bus error syndromes

Syndrome	Description	Direction
SYN_ACE_BAD	ACE5-Lite transactions are either bad or unrecognized.	Read and write
SYN_GICD_CORRUPTED	Data read from SPI RAM is corrupted.	Read-only
SYN_GICR_CORRUPTED	Data read from SGI or PPI RAM is corrupted.	Read-only
SYN_ITS_OFF	Access to ITS attempted when powered down.	Read and write

5. Programmers model for GIC-720AE

All the GIC-720AE registers have names that are constructed of mnemonics that indicate the logical block that the register belongs to and the register function.

The following information applies to the GIC-720AE registers:

- The GIC-720AE implements only memory-mapped registers.
- The GIC-720AE has a single base address, except for the GITS_TRANSLATER register. The base address is not fixed and can be different for each particular system implementation.
- The offset of each register from the base address is fixed.
- Accesses to reserved or unused address locations might result in a bus error, depending on the value of GICT_ERR0CTLR.UE and GICT_ERR0CTLR.DIS_ACE.
- Unless otherwise stated in the accompanying text:
  - Do not modify reserved register bits.
  - Ignore reserved register bits on reads.
  - A system reset or a Cold reset, resets all register bits to zero.
- The GIC-720AE ACE5-Lite subordinate interface can be 64 bits, 128 bits, 256 bits, or 512 bits wide, depending on the configuration. The *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* defines the permitted sizes of access.


Note

The GIC-720AE guarantees single-copy atomicity for doubleword accesses.

- The GIC-720AE supports data only in little-endian format.
- The access types for the GIC-720AE are as follows:

RO	Read-only
RW	Read and write
WO	Write-only, reads return as **UNKNOWN**.

- Unless specified otherwise, all Secure registers are accessible by Non-secure accesses when security is disabled, that is, GICD_CTLR.DS == 1.

5.1 Register map pages

The GIC-720AE address map has multiple pages. The number of pages and the address aliasing depends on the GIC configuration.

The registers in the following table are accessible through the ACE5-Lite interface. For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, then each view sees the register map pages that the table shows.

The *Fault Management Unit* (FMU) registers are accessible through the APB5 interface. See:

- 5.12 FMU register summary on page 291
- 6.1.1 FMU APB5 interface on page 334

Table 5-1: Register map pages

Page offset		Page	Description
No v4.1 support	With v4.1 support		
0		GICD	GICD main page
1		GICM	GICM message-based interrupts
2		GICT	GIC trace and debug page
3		GICP	GIC PMU page
4 + 2×ITSnum	4 + 4×ITSnum	GITS	ITS address page. ITSnum is the serial number of each ITS, which is from 0 to ITScount−1.
5 + 2×ITSnum	5 + 4×ITSnum	GITS (translate)	ITS translation page
6 + 2×ITSnum	6 + 4×ITSnum	GITS (vSGI)	ITS vSGI page
7 + 2×ITSnum	7 + 4×ITSnum	Reserved	Reserved
4 + 2×ITScount + 2×RDnum	4 + 4×ITScount + 4×RDnum	GICR (LPI)	GICR LPI registers. ITScount is the total number of ITS.
5 + 2×ITScount + 2×RDnum	5 + 4×ITScount + 4×RDnum	GICR (SGI)	GICR PPI + SGI registers. RDnum is the serial number of each “internal Redistributor”, which is from 0 to RDcount−1.
6 + 2×ITScount + 2×RDnum	6 + 4×ITScount + 4×RDnum	GICR (vLPI)	GICR vLPI registers
7 + 2×ITScount + 2×RDnum	7 + 4×ITScount + 4×RDnum	Reserved	Reserved
4 + 2×ITScount + 2×RDcount	4 + 4×ITScount + 4×RDcount	GICDA	Alias to GICD (page after last GICR page). RDcount is the total number of “internal Redistributors”, which equals total number of CPU cores. RDcount can change if the GICD_RDOFFRn registers or the gicd_pe_off tie-off signal removes Redistributors. In this case, the GICDA page moves to the page above the last Redistributor.

For more information, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

You must set up the system address map so that each core accesses the GICD page on its local chip at the same address. All other pages must be globally accessible, although access of pages on a remote chip by a core is expected to be rare. Allowing the GIC pages to be globally accessible might require the system interconnect to alias the page addresses.

Page offset

The ACE5-Lite address bits[x:16] control which GIC register page is accessed in Table 5-1: Register map pages on page 133. The value of x depends on the axis_addr_width GICD configuration parameter.

In non-monolithic configurations, the GIC-720AE ignores address bits above ceil[$\log_2$(page_count)] + 15. For example, a configuration that uses 11 pages ignores address bits above 19, so any address bits of the form 0xXXXXX00000 is accepted and it accesses the GICD page.

In monolithic configurations, where the Distributor and ITS share the ACE5-Lite subordinate port, the gicd_page_offset and its_transr_page_offset address tie-off signals control the full page address of the GICD and GITS_TRANSLATER pages. The page address comprises address bits[x:16]. For example, if the GICD page is at 32-bit address 0xFFFF0000, the gicd_page_offset tie-off is 16-bit 0xFFFF.

In multi view configurations, the two address bits that are higher than those that identify all the pages in a particular view are used as a view identifier. For example, if a view has 16 pages, then it requires 4 bits to address those pages. Therefore, address bits[21:20] are used to select a view.

5.1.1 Discovery

We recommend that the operating system is provided with pointers to the start of the Distributor, every ITS, and the first Redistributor page on each chip.

To verify that the pages relate to GIC registers, software can check these pointers against the discovery registers, which start at offset 0xFFD0 for each GIC page. These registers allow discovery of the architecture version and, for GIC-720AE, whether the page contains the Distributor, ITS, or Redistributor registers. For example, to discover the page type, software can:

1. Read from 0xFFE0 to determine the PIDR0.PART_0 value.
2. Read from 0xFFE4 to determine the PIDR1.PART_1 value.
3. Concatenate PART_1 (4 bits) and PART_0 (8 bits), to discover the 12-bit part number, PART_1|| PART_0. A value of:
   - 0x492 indicates that this page contains Distributor registers.
   - 0x493 indicates that this page contains Redistributor registers.
   - 0x494 indicates that this page contains ITS registers.

When this information is known, software can obtain additional information from registers that are specific to each page.

For Redistributors, we recommend that you examine GICR_TYPER to determine:

- Whether the implementation has two or four pages for each Redistributor, which depends on the features implemented. It can be inferred that GIC-720AE has four pages for each Redistributor because the GICR_TYPER.VLPIS bit indicates that it supports virtual LPIs.

- Whether it is the last Redistributor in the series of pages. However, if GICD_CFGID.VIEW == 1, then it applies only for view 0 because GICR_TYPER.Last == 1 for views 1, 2, and 3.
- Which core the Redistributor is for, based on affinity values.

This information allows you to iteratively search through all Redistributors in a discovery process.

The GITS_TYPER register in the GIC-720AE indicates that you must program the ITS with unique ProcessorNumbers, instead of physical target addresses. The GICR_TYPER contains the unique ProcessorNumber that you must use to reference a Redistributor when programming the ITS.

Note

In a multichip configuration, the ProcessorNumber upper bits are derived from the chip_id tie-off signal. Therefore, the chip_id signal value must be set before the GIC exits from reset.

For more information, see the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4*.

5.1.2 GIC-720AE register access and banking

The GIC-720AE uses an access and banking scheme for its registers.

For more information about the register access and banking scheme, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

The key characteristics of the scheme are:

- Some registers such as the *Distributor Control Register*, GICD_CTLR, and the *Redistributor Control Register*, GICR_CTLR, are banked by security that provides separate Secure and Non-secure copies of the registers. A Secure access to the address, accesses the Secure copy of the register. A Non-secure access to the address, accesses the Non-secure copy.
- Some registers, such as the *Interrupt Group Registers*, GICD_IGROUPRn, are only accessible using Secure accesses.
- Non-secure accesses to registers, or parts of a register, which are only accessible to Secure accesses are *Read-As-Zero* and *Writes Ignored* (RAZ/WI).

5.2 Distributor registers (GICD/GICDA) summary

The GIC-720AE Distributor functions are controlled through the Distributor registers identified with the prefix GICD. The Distributor Alias registers are identified with the prefix GICDA.

The following table lists the Distributor registers in base offset order and provides a reference to the register description that is described in either this document or the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Address offsets are relative to the Distributor base address defined by the system memory map.

Offsets that are not shown or are marked as reserved, are Reserved and RAZ/WI. Accesses to these offsets might be reported in error record 0 as a SYN_ACE_BAD access.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, the accessibility of a GICD register depends on the view. See Multi view access to the GICD registers on page 140 for information about the views that a register is accessible in.

Table 5-2: Distributor registers (GICD/GICDA) summary

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0000	GICD_CTLR	RW	Configuration dependent	32	Distributor Control Register	Yes
0x0004	GICD_TYPER	RO	Configuration dependent	32	Interrupt Controller Type Register	Yes
0x0008	GICD_IIDR	RO	0x070nn43B The nn value depends on the *rxpy* identifier.	32	Distributor Implementer Identification Register	Yes
0x000C	GICD_TYPER2	RO	Configuration dependent	32	Interrupt Controller Type 2 Register	Yes
0x0010-0x001C	-	-	-	-	Reserved	-
0x0020	GICD_FCTLR	RW	0x0	32	Function Control Register	No
0x0024	GICD_SAC	RW	Tie-off signal dependent	32	Secure Access Control register	No
0x0028	GICD_CCCGR	RW	0x0	32	Cross-Chip Control Group Register. Only present for multichip configurations when GICD_CFGID.ACE_CC == 0.	No
0x002C	GICD_CCCCR	RW	0x0	32	Cross-Chip Control Credit Register. Only present for multichip configurations when GICD_CFGID.ACE_CC == 0.	No
0x0030	GICD_FCTLR2	RW	0x0	32	Function Control Register 2	No
0x0034	GICD_UTILR	RW	0x0	32	Utilization Register	No
0x0038	GICD_FCTLR3	RW	0x9F	32	Function Control Register 3	No
0x003C	GICD_CCCTLR	RW	0x0	32	Cross-Chip Control Register. Only present when GICD_CFGID.ACE_CC == 1.	No
0x0040	GICD_SETSPI_NSR	WO	-	32	Non-secure SPI Set Register	Yes
0x0044	-	-	-	-	Reserved	-
0x0048	GICD_CLRSPI_NSR	WO	-	32	Non-secure SPI Clear Register	Yes
0x004C	-	-	-	-	Reserved	-
0x0050	GICD_SETSPI_SR	WO	-	32	Secure SPI Set Register. Only present when Security support is included, otherwise Reserved.	Yes
0x0054	-	-	-	-	Reserved	-
0x0058	GICD_CLRSPI_SR	WO	-	32	Secure SPI Clear Register. Only present when Security support is included, otherwise Reserved.	Yes
0x005C-0x007C	-	-	-	-	Reserved	-

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0080-0x00FC	GICD_IGROUPRn	RW	0x0	32	Interrupt Group Registers, n = 0-31, but n=0 is Reserved	Yes
0x0100-0x017C	GICD_ISENABLERn	RW	0x0	32	Interrupt Set-Enable Registers, n = 0-31, but n=0 is Reserved	Yes
0x0180-0x01FC	GICD_ICENABLERn	RW	0x0	32	Interrupt Clear-Enable Registers, n = 0-31, but n=0 is Reserved	Yes
0x0200-0x027C	GICD_ISPENDRn	RW	SPI signal dependent	32	Interrupt Set-Pending Registers, n = 0-31, but n=0 is Reserved	Yes
0x0280-0x02FC	GICD_ICPENDRn	RW	SPI signal dependent	32	Interrupt Clear-Pending Registers, n = 0-31, but n=0 is Reserved	Yes
0x0300-0x037C	GICD_ISACTIVERn	RW	0x0	32	Interrupt Set-Active Registers, n = 0-31, but n=0 is Reserved	Yes
0x0380-0x03FC	GICD_ICACTIVERn	RW	0x0	32	Interrupt Clear-Active Registers, n = 0-31, but n=0 is Reserved	Yes
0x0400-0x07FC	GICD_IPRIORITYRn	RW	Security dependent	32	Interrupt Priority Registers, n = 0-255, but n=0-7 are Reserved when affinity routing is enabled	Yes
0x0800-0x0BFC	-	-	-	-	Reserved	-
0x0C00-0x0CFC	GICD_ICFGRn	RW	0x0	32	Interrupt Configuration Registers, n = 0-63, but n=0-1 are Reserved	Yes
0x0D00-0x0D7C	GICD_IGRPMODRn	RW	0x0	32	Interrupt Group Modifier Registers, n = 0-31, but n=0 is Reserved. If GICD_CTLR.DS == 1, then this register is RAZ/WI.	Yes
0x0D80-0x0DFC	-	-	-	-	Reserved	-
0x0E00-0x0EFC	GICD_NSACRn	RW	0x0	32	Non-secure Access Control Registers, n = 0-63, but n=0-1 are Reserved when affinity routing is enabled. Only present when Security support is included, otherwise Reserved.	Yes
0x0F00-0x0FFC	-	-	-	-	Reserved	-
0x1000-0x107C	GICD_IGROUPRnE	RW	0x0	32	Interrupt Group Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x1080-0x11FC	-	-	-	-	Reserved	-
0x1200-0x127C	GICD_ISENABLERnE	RW	0x0	32	Interrupt Set-Enable Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x1280-0x13FC	-	-	-	-	Reserved	-
0x1400-0x147C	GICD_ICENABLERnE	RW	0x0	32	Interrupt Clear-Enable Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x1480-0x15FC	-	-	-	-	Reserved	-
0x1600-0x167C	GICD_ISPENDRnE	RW	SPI signal dependent	32	Interrupt Set-Pending Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x1680-0x17FC	-	-	-	-	Reserved	-

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x1800-0x187C	GICD_ICPENDRnE	RW	SPI signal dependent	32	Interrupt Clear-Pending Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x1880-0x19FC	-	-	-	-	Reserved	-
0x1A00-0x1A7C	GICD_ISACTIVERnE	RW	0x0	32	Interrupt Set-Active Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x1A80-0x1BFC	-	-	-	-	Reserved	-
0x1C00-0x1C7C	GICD_ICACTIVERnE	RW	0x0	32	Interrupt Clear-Active Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x1C80-0x1FFC	-	-	-	-	Reserved	-
0x2000-0x23FC	GICD_IPRIORITYRnE	RW	0x0	32	Interrupt Priority Registers Extended, n = 0-255. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x2400-0x2FFC	-	-	-	-	Reserved	-
0x3000-0x30FC	GICD_ICFGRnE	RW	0x0	32	Interrupt Configuration Registers Extended, n = 0-63. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x3100-0x33FC	-	-	-	-	Reserved	-
0x3400-0x347C	GICD_IGRPMODRnE	RW	0x0	32	Interrupt Group Modifier Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved. If GICD_CTLR.DS == 1, then this register is RAZ/WI.	Yes
0x3480-0x35FC	-	-	-	-	Reserved	-
0x3600-0x36FC	GICD_NSACRnE	RW	0x0	32	Non-secure Access Control Registers Extended, n = 0-63. Only present when > 960 SPIs, otherwise Reserved.	Yes
0x3700-0x5FFC	-	-	-	-	Reserved	-
0x6000-0x7FF8	GICD_IROUTERn	RW	0x0080000000 if configured.	64	Interrupt Routing Registers, n = 0-991, but n=0-31 are Reserved when affinity routing is enabled. See the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4*. All SPIs are reset with Interrupt_Routing_Mode == 1. The first register is GICD_IROUTER32, at address 0x6100.	Yes
0x8000-0x9FF8	GICD_IROUTERnE	RW	0x0	64	Interrupt Routing Registers Extended, n = 0-1023. Only present when > 960 SPIs, otherwise Reserved.	Yes
0xA000-0xBFFC	-	-	-	-	Reserved	-
0xC000	GICD_CHIPSR	RO	P-Channel dependent	32	Chip Status Register. Reserved in single-chip configurations.	No
0xC004	GICD_DCHIPR	RW	0x0	32	Default Chip Register. Reserved in single-chip configurations.	No
0xC008-0xC080	GICD_CHIPRn	RW	0x0	64	Chip Registers, n = 0-15. Reserved in single-chip configurations.	No

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0xC088-0xC7FC	-	-	-	-	Reserved	-
0xC800-0xC838	GICD_RDOFFRn	RW	0x0	64	Redistributor Off Registers, n = 0-7. Only present when GICD_CFGID.RDC == 1.	No
0xC840-0xD014	-	-	-	-	Reserved	-
0xD018	GICD_VCFGBASER	RO	0x0	64	Copy of GICR_VCFGBASER. Only present when no local redistributors.	No
0xD020-0xD05C	-	-	-	-	Reserved	-
0xD060	GICD_VSLEEPR	RW	0x0	32	vICM Sleep Register. Only present when no local redistributors.	No
0xD064-0xDFFC	-	-	-	-	Reserved	-
0xE000-0xE0FC	GICD_ICLARn	RW	0x0	32	Interrupt Class Registers, n = 0-63, but n=0-1 are Reserved	No
0xE100-0xE17C	GICD_ICERRRn	RW	0x0	32	Interrupt Clear Error Registers, n = 0-31, but n=0 is Reserved	No
0xE180-0xE1FC	GICD_ICGERRn	RW	0x0	32	Interrupt Clear Group Error registers, n = 0-31, but n=0 is Reserved	No
0xE200-0xE27C	GICD_ISERRRn	RW	0x0	32	Interrupt Set Error Registers, n = 0-31, but n=0 is Reserved	No
0xE280-0xE2FC	-	-	-	-	Reserved	-
0xE400-0xE47C	GICD_ICERRRnE	RW	0x0	32	Interrupt Clear Error Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	No
0xE480-0xE5FC	-	-	-	-	Reserved	-
0xE600-0xE67C	GICD_ICGERRnE	RW	0x0	32	Interrupt Clear Group Error registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	No
0xE680-0xE7FC	-	-	-	-	Reserved	-
0xE800-0xE87C	GICD_ISERRRnE	RW	0x0	32	Interrupt Set Error Registers Extended, n = 0-31. Only present when > 960 SPIs, otherwise Reserved.	No
0xE880-0xE9FC	-	-	-	-	Reserved	-
0xEA00-0xEA78	GICD_ERRINSRn	RW	Configuration dependent	64	Error Insertion Registers, n = 0-15	No
0xEA78-0xEBFC	-	-	-	-	Reserved	-
0xEC00-0xECFC	GICD_ICLARnE	RW	0x0	32	Interrupt Class Registers Extended, n = 0-63. Only present when > 960 SPIs, otherwise Reserved.	No
0xED00-0xEFFC	-	-	-	-	Reserved	-
0xF000	GICD_CFGID	RO	Configuration dependent	64	Configuration ID Register	No

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0xF008-0xF1FC	-	-	-	-	Reserved	-
0xF200-0xF27C	GICD_ICVERRRn	RW	0x0	32	Interrupt Clear View Error Registers	No
0xF280-0xF3FC	-	-	-	-	Reserved	-
0xF400-0xF47C	GICD_ICVERRRnE	RW	0x0	32	Interrupt Clear View Error Registers Extended	No
0xF480-0xF5FC	-	-	-	-	Reserved	-
0xF600-0xF6FC	GICD_IVIEWRn	RW	0x0	32	Interrupt View Registers	No
0xF700-0xF7FC	-	-	-	-	Reserved	-
0xF800-0xF8FC	GICD_IVIEWRnE	RW	0x0	32	Interrupt View Registers Extended	No
0xF900-0xFFCC	-	-	-	-	Reserved	-
0xFFD0	GICD_PIDR4	RO	0x44	32	Peripheral ID 4 Register	Yes
0xFFD4	GICD_PIDR5	RO	0x00	32	Peripheral ID 5 Register	Yes
0xFFD8	GICD_PIDR6	RO	0x00	32	Peripheral ID 6 Register	Yes
0xFFDC	GICD_PIDR7	RO	0x00	32	Peripheral ID 7 Register	Yes
0xFFE0	GICD_PIDR0	RO	0x92	32	Peripheral ID 0 Register	Yes
0xFFE4	GICD_PIDR1	RO	0xB4	32	Peripheral ID 1 Register	Yes
0xFFE8	GICD_PIDR2	RO	Configuration dependent	32	Peripheral ID 2 Register	Yes
0xFFEC	GICD_PIDR3	RO	0x00	32	Peripheral ID 3 Register	Yes
0xFFF0	GICD_CIDR0	RO	0x0D	32	Component ID 0 Register	Yes
0xFFF4	GICD_CIDR1	RO	0xF0	32	Component ID 1 Register	Yes
0xFFF8	GICD_CIDR2	RO	0x05	32	Component ID 2 Register	Yes
0xFFFC	GICD_CIDR3	RO	0xB1	32	Component ID 3 Register	Yes

Multi view access to the GICD registers

The following table shows the views that a GICD register is accessible in.

Table 5-3: Multi view access to the Distributor registers

Name	View
GICD_CTLR	Banked
GICD_TYPER	Banked
GICD_IIDR	0, 1, 2, 3
GICD_TYPER2	0, 1
GICD_FCTLR	0
GICD_SAC	0

Name	View
GICD_CCCGR	0
GICD_CCCCR	0
GICD_FCTLR2	See register for more information.
GICD_UTILR	0, 1
GICD_FCTLR3	0
GICD_CCCTLR	0
GICD_SETSPI_NSR	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_CLRSPI_NSR	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_SETSPI_SR	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_CLRSPI_SR	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_IGROUPRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ISENABLERn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICENABLERn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ISPENDRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICPENDRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ISACTIVERn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICACTIVERn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_IPRIORITYRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICFGRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_IGRPMODRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_NSACRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_IGROUPRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ISENABLERnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICENABLERnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ISPENDRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICPENDRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ISACTIVERnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICACTIVERnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_IPRIORITYRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICFGRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_IGRPMODRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_NSACRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_IROUTERn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_IROUTERnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_CHIPSR	0
GICD_DCHIPR	0
GICD_CHIPRn	0
GICD_RDOFFRn	0
GICD_VCFGBASER	0, 1
GICD_VSLEEPR	0, 1

Name	View
GICD_ICLARn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICERRRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICGERRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ISERRRn	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICERRRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ICGERRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ISERRRnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_ERRINSRn	0
GICD_ICLARnE	0 and the view that GICD_IVIEWRn or GICD_IVIEWRnE sets.
GICD_CFGID	Banked, see register for more information.
GICD_ICVERRRn	0
GICD_ICVERRRnE	0
GICD_IVIEWRn	0
GICD_IVIEWRnE	0
GICD_PIDR4	0, 1, 2, 3
GICD_PIDR5	0, 1, 2, 3
GICD_PIDR6	0, 1, 2, 3
GICD_PIDR7	0, 1, 2, 3
GICD_PIDR0	0, 1, 2, 3
GICD_PIDR1	0, 1, 2, 3
GICD_PIDR2	0, 1, 2, 3
GICD_PIDR3	0, 1, 2, 3
GICD_CIDR0	0, 1, 2, 3
GICD_CIDR1	0, 1, 2, 3
GICD_CIDR2	0, 1, 2, 3
GICD_CIDR3	0, 1, 2, 3

5.2.1 GICD_CTLR, Distributor Control Register

This register enables interrupts for Group 0 and Group 1. It also indicates whether the Distributor supports one or two security states, and whether a register write is in progress.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, this register is banked for each of the views.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for the different architectural views of the GICD_CTLR register.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-1: GICD_CTLR bit assignments

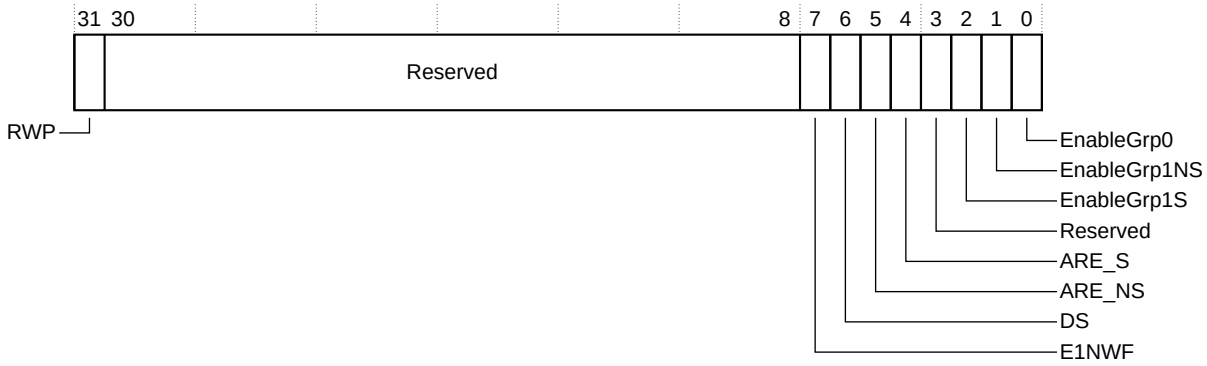


Table 5-4: GICD_CTLR bit descriptions

Bits	Name	Description	Type	Reset
[31]	RWP	Register Write Pending: **0** No register write in progress. **1** Register write in progress.	RO	0
[30:8]	-	Reserved	-	0
[7]	E1NWF	Enable 1 of N Wakeup Functionality	RW	0
[6]	DS	Disable Security status: **0** The gicd_ctlr_ds signal was LOW when the GIC exited reset. Therefore, the Distributor supports two Security states and Non-secure accesses cannot access and modify registers that control Group 0 interrupts. **1** The gicd_ctlr_ds signal was HIGH when the GIC exited reset. Therefore, the Distributor only supports a single Security state and Non-secure accesses can access and modify registers that control Group 0 interrupts. See 4.3 Interrupt groups and security on page 59 for more information.	RO	gicd_ctlr_ds signal
[5]	ARE_NS	Affinity Routing Enable, Non-secure state. This bit is RES0 when GICD_CTLR.DS == 1.	RO	1
[4]	ARE_S	Affinity Routing Enable, Secure state. However, if GICD_CTLR.DS == 1, this bit is ARE and applies to the single security state.	RO	1
[3]	-	Reserved	-	0
[2]	EnableGrp1S	Enable Secure Group 1 interrupts. This bit is RES0 when GICD_CTLR.DS == 1.	RW	0

Bits	Name	Description	Type	Reset
[1]	EnableGrp1NS	Enable Non-secure Group 1 interrupts. However, if GICD_CTLR.DS == 1, enable Group 1 interrupts.	RW	0
[0]	EnableGrp0	Enable Group 0 interrupts	RW	0

5.2.2 GICD_TYPER, Interrupt Controller Type Register

This register returns information about the configuration of the GIC-720AE. You can use this register to determine the number of Security states, the number of INTIDs, and the number of processor cores that the GIC supports.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, this register is banked for each of the views.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-2: GICD_TYPER bit assignments

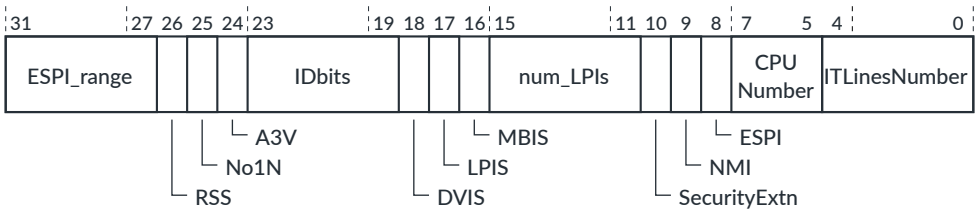


Table 5-5: GICD_TYPER bit assignments

Bits	Name	Description
[31:27]	ESPI_Range	Returns the number of extended SPIs that GIC-720AE supports, and is given by 32×`spi_blocks` – 960. The `spi_blocks` parameter is set when the GIC is configured.
[26]	RSS	Range selector support. Returns: **0** The GIC supports targeted SGIs with affinity level 0 values of 0-15.

Bits	Name	Description
[25]	No1N	1 of N SPI: **0** The GIC-720AE supports 1 of N SPI interrupts. This value occurs when `spi_1ofn_support` == 1. **1** The GIC-720AE does not support 1 of N SPI interrupts. This value occurs when `spi_1ofn_support` == 0.
[24]	A3V	Affinity level 3 values. Depending on the configuration, returns either: **0** The GIC-720AE Distributor supports only zero values of affinity level 3. **1** The GIC-720AE Distributor supports nonzero values of affinity level 3.
[23:19]	IDbits	Interrupt identifier bits: **`0b01111`** The GIC-720AE supports 16 interrupt identifier bits.
[18]	DVIS	Direct virtual LPI injection support: **0** The GIC-720AE does not support direct virtual LPI injection. **1** The GIC-720AE does support direct virtual LPI injection. See the *Arm® Generic Interrupt Controller v3 and v4 - Virtualization*.
[17]	LPIS	Indicates whether the GIC supports LPIs. Depending on the configuration, returns either: **0** LPIs are not supported. **1** LPIs are supported.
[16]	MBIS	Message-based interrupt support: **1** The GIC-720AE supports message-based interrupts.
[15:11]	num_LPIs	Returns `0b00000` because GICD_TYPER.IDbits indicates the number of LPIs that the GIC supports.
[10]	SecurityExtn	Security state support. Depending on the gicd_ctlr_ds signal as the GIC exits reset, returns either: **0** gicd_ctlr_ds signal was HIGH during reset, so the GIC-720AE supports only a single Security state. **1** gicd_ctlr_ds signal was LOW during reset, so the GIC-720AE supports two Security states.
[9]	NMI	Indicates whether the GIC supports non-maskable interrupts (NMIs). Depending on the configuration, returns either: **0** NMIs are not supported. This value occurs when `nmi_support` == 0. **1** NMIs are supported. This value occurs when `nmi_support` == 1.
[8]	ESPI	Extended SPI: **0** The GIC is configured to support ≤960 SPIs. **1** The GIC is configured to support >960 SPIs.
[7:5]	CPUNumber	Returns `0b000` because GICD_CTLR.ARE==1 (ARE_NS & ARE_S).
[4:0]	ITLinesNumber	Returns the maximum SPI INTID that this GIC-720AE implementation supports, and is given by 32×(ITLinesNumber + 1) − 1. If GICD_TYPER.ESPI ==1, then this field returns `0x1E`.

5.2.3 GICD_IIDR, Distributor Implementer Identification Register

This register provides information about the implementer and revision of the Distributor.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-3: GICD_IIDR bit assignments

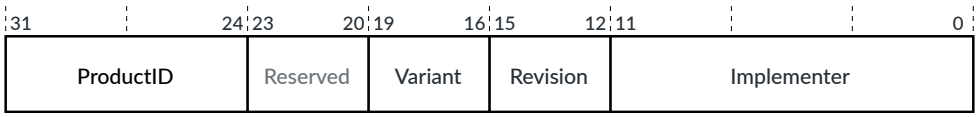


Table 5-6: GICD_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **0x07** GIC-720AE
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: **0x0** r0 **0x1** r1 **0x2** r2
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: **0x0** p0 **0x1** p1
[11:0]	Implementer	Identifies the implementer: **0x43B** Arm

5.2.4 GICD_TYPER2, Interrupt Controller Type Register 2

This register returns the number of bits that GIC-720AE uses for a vPEID.

Configurations

This register is available in all configurations.

Attributes

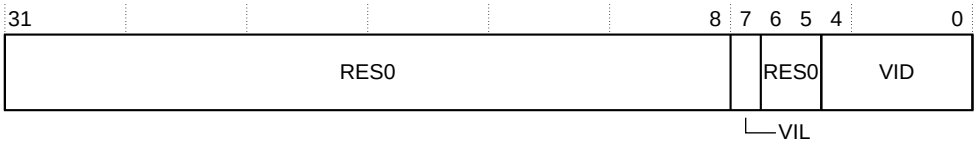
Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-4: GICD_TYPER2 bit assignments



In the following table, the View column is applicable only for GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1.

Table 5-7: GICD_TYPER2 bit descriptions

Bits	Name	Description	View
[31:8]	-	Reserved, **RES0**.	-
[7]	VIL	Returns the number of bits that GIC-720AE can use for a vPEID: **0** GIC-720AE supports 16 bits of vPEID. **1** GIC-720AE supports GICD_TYPER2.VID + 1 bit of vPEID. If GICD_TYPER.DVIS == 0, then this bit returns zero.	0 and 1 only. Returns zero for views 2 and 3.
[6:5]	-	Reserved, **RES0**.	-
[4:0]	VID	Returns the value of the `vpe_width` configuration parameter. Values above `0xF` are reserved. If GICD_TYPER.DVIS == 0, then this field returns zero.	0 and 1 only. Returns zero for views 2 and 3.

5.2.5 GICD_FCTLR, Function Control Register

This register controls non-architectural functionality such as the scrubbing of all RAMs in the local Distributor. The register is not distributed and acts only on the local chip.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-5: GICD_FCTLR bit assignments

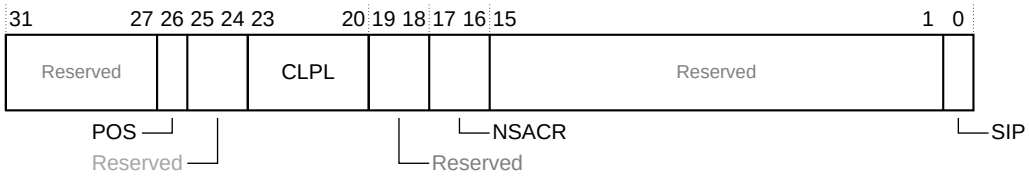


Table 5-8: GICD_FCTLR bit assignments

Bits	Name	Description
[31:27]	-	Reserved, RES0
[26]	POS	Point of serialization. Secure access only. When an interrupt is sent remotely and POS is set, it ensures that writes to GICD_SETSPI and GICD_CLRSPI propagate to remote chips before ACE5-Lite sends a response. Applies only to edge-triggered interrupts. **0** Store locally and propagate when possible. **1** Propagate access to POS. Resets to `0b0`.
[25:24]	-	Reserved, RES0
[23:20]	CLPL	Cross-chip LPI limit. Secure access only. This field enables you to reduce the number of cross-chip LPI transactions that can be outstanding to each chip: **0** The `lpi_cc_tokens` configuration parameter sets the maximum number of cross-chip LPI transactions that can be outstanding to each chip. **1-15** The maximum number of cross-chip LPI transactions that can be outstanding to each chip. If you set a value that is greater than `lpi_cc_tokens`, then the GIC behaves as if CLPL == 0.
[19:18]	-	Reserved, RES0

Bits	Name	Description
[17:16]	NSACR	Non-secure access control. Values are as described in the GICD_NSACR register. This is the value that is used if an SPI has an error. Secure access only. Resets to `0b00`.
[15:1]	-	Reserved, returns `0b000`
[0]	SIP	Scrub in progress. When read: **0** No scrub in progress. **1** Scrub in progress. When written: **0** Abort the scrub. **1** Start a scrub. When a scrub is complete, the GIC clears the bit to 0.

Accessibility

If GICD_CFGID.VIEW == 0, some bits are accessible only by Secure accesses. If GICD_CFGID.VIEW == 1, then GICD_FCTLR is accessible only by Secure accesses from view 0.

5.2.6 GICD_SAC, Secure Access Control register

This register allows Secure software to grant Non-secure software with access to some GIC-720AE Secure features. It also controls whether Secure PMU events are visible to Non-secure software. For configurations that support multi view, it controls which view can access the GICP registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-6: GICD_SAC bit assignments

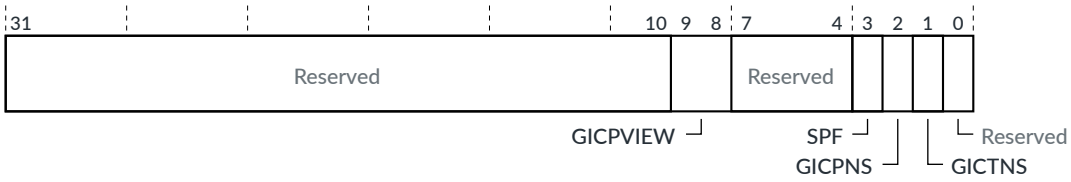


Table 5-9: GICD_SAC bit assignments

Bits	Name	Description	Type
[31:10]	-	Reserved, returns zero	-
[9:8]	GICPVIEW	For configurations that support multi view, that is, when GICD_CFGID.VIEW == 1, this field controls which view can access the GICP registers: **0** Only view 0 can access the GICP registers. The PMU records all events. This value occurs at reset. **1** View 0 and view 1 can access the GICP registers. The PMU records events from view 1 only. **2** View 0 and view 2 can access the GICP registers. The PMU records events from view 2 only. **3** View 0 and view 3 can access the GICP registers. The PMU records events from view 3 only. RAZ/WI when no multi view support.	RW
[7:4]	-	Reserved, returns zero	-
[3]	SPF	Controls whether Secure PMU events are visible to Non-secure software: **0** Secure PMU event masking is disabled. The GIC reports Secure and Non-secure PMU events to Non-secure software and Secure software. This value occurs at reset. **1** Secure PMU event masking is enabled. The GIC reports Non-secure PMU events but it does not report Secure PMU events to Non-secure software. All PMU events are visible to Secure software. When GICD_CFGID.VIEW == 1, if GICPVIEW == 0, then do not set SPF to 1.	RW
[2]	GICPNS	Controls whether the Non-secure world can access the Secure PMU data: **0** Secure access only to the GICP registers. **1** Allow Non-secure access to the GICP registers. The gicp_allow_ns tie-off signal controls the reset value for each chip.	RW
[1]	GICTNS	Controls whether the Non-secure world can access the Secure trace data and the error insertion registers: **0** Secure access only to the GICT registers and the error insertion registers. **1** Allow Non-secure access to the GICT registers and the error insertion registers. The error insertion registers are GICD_ERRINSRn, GICR_ERRINSR, GITS_D_ERRINSR, GITS_V_ERRINSR, and GITS_C_ERRINSR. The gict_allow_ns tie-off signal controls the reset value for each chip.	RW
[0]	-	Reserved, RES0	-

Accessibility

GICD_SAC is accessible only by Secure accesses.

5.2.7 GICD_CCCGR, Cross-Chip Control Group Register

This register enables software to assign each chip to one of four credit groups. A credit group sets the number of outstanding AXI5-Stream transactions that can be sent to that group of chips.

Configurations

This register is available in multichip configurations when GICD_CFGID.ACE_CC == 0. RES0 when GICD_CFGID.ACE_CC == 1.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-7: GICD_CCCGR bit assignments

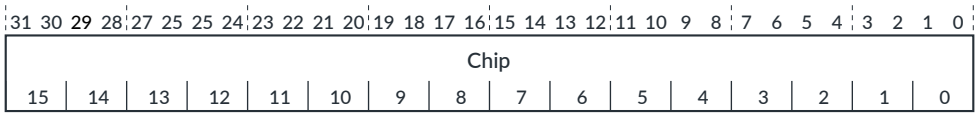


Table 5-10: GICD_CCCGR bit descriptions

Bits	Name	Description
[31:0] Bits[2n+1:2n], for n = 0 to 15	Chip<n>	Controls the credit group that software assigns to chip n: **0b00** Chip n is in credit group 0, which supports GICD_CCCCR.Group0 outstanding AXI5-Stream transactions. **0b01** Chip n is in credit group 1, which supports GICD_CCCCR.Group1 outstanding AXI5-Stream transactions. **0b10** Chip n is in credit group 2, which supports GICD_CCCCR.Group2 outstanding AXI5-Stream transactions. **0b11** Chip n is in credit group 3, which supports GICD_CCCCR.Group3 outstanding AXI5-Stream transactions. The chip_id tie-off signal sets the value of n for each chip.

Accessibility

GICD_CCCGR is accessible only by Secure accesses.

5.2.8 GICD_CCCCR, Cross-Chip Control Credit Register

This register controls the number of outstanding AXI5-Stream transactions to a set of remote chips that are assigned to the same credit group. The GICD_CCCGR register controls the assignment of chips to a credit group.

Configurations

This register is available in multichip configurations when GICD_CFGID.ACE_CC == 0. RES0 when GICD_CFGID.ACE_CC == 1.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-8: GICD_CCCCR bit assignments

Table 5-11: GICD_CCCCR bit descriptions

Bits	Name	Description
[31:24]	Group3	The number of outstanding AXI5-Stream transactions that are available for chips that GICD_CCCGR assigns to group 3: **n** n outstanding AXI5-Stream transactions available. **0** No limit.
[23:16]	Group2	The number of outstanding AXI5-Stream transactions that are available for chips that GICD_CCCGR assigns to group 2: **n** n outstanding AXI5-Stream transactions available. **0** No limit.
[15:8]	Group1	The number of outstanding AXI5-Stream transactions that are available for chips that GICD_CCCGR assigns to group 1: **n** n outstanding AXI5-Stream transactions available. **0** No limit.
[7:0]	Group0	The number of outstanding AXI5-Stream transactions that are available for chips that GICD_CCCGR assigns to group 0: **n** n outstanding AXI5-Stream transactions available. **0** No limit.

Accessibility

GICD_CCCCR is accessible only by Secure accesses.

5.2.9 GICD_FCTLR2, Function Control Register 2

This register controls clock gating and other non-architectural controls in the local Distributor. The register is not distributed and acts only on the local chip.

Configurations

This register is available in all configurations.

Attributes

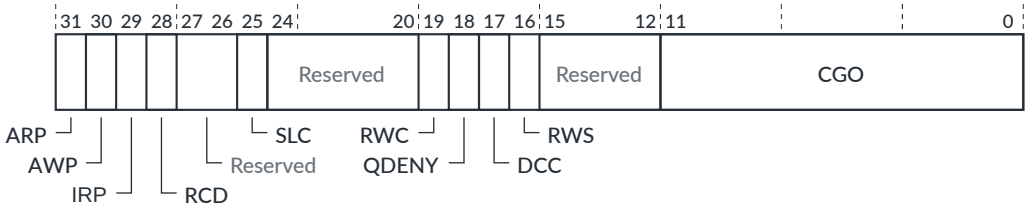
Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-9: GICD_FCTLR2 bit assignments



In the following table, the View column is applicable only for GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1.

Table 5-12: GICD_FCTLR2 bit assignments

Bits	Name	Description	View
[31]	ARP	Report read poison if corrupted data from a RAM is read.	0
[30]	AWP	Report write poison. Reject poisoned writes on the subordinate interface.	0
[29]	IRP	Ignore read poison from manager.	0
[28]	RCD	Read chunking disable.	0
[27:26]	-	Reserved, RES0	0
[25]	SLC	Strict LPI caching: **0** Use fully associative caching in the LPI caches. We recommend that SLC == 0, to use fully associative caching for LPIs. **1** Use 2-way set associative caching in the LPI caches.	0, 1

Bits	Name	Description	View
[24:20]	-	Reserved, RES0	-
[19]	RWC	Residency wait on command. See 4.7.2 Residency and VMOVP on page 68 for more information.	0, 1
[18]	QDENY	Q-Channel deny. Overrides the Q-Channel logic and forces the Distributor to reject powerdown requests.	0
[17]	DCC	Do not correct cache. Modifies the a<x>cache output signal from the Distributor. See 4.14 Memory access and attributes on page 89.	0, 1
[16]	RWS	Residency wait on *Pending Table System* (PTS) RAM search. See 4.7.2 Residency and VMOVP on page 68 for more information.	0, 1
[15:12]	-	Reserved, RES0	-
[11:0]	CGO	Clock gate override. One bit for each clock gate: **0** Use full clock gating. **1** Leave clock running. If clock gates are not implemented, then you must use this value. The clock gate bit assignments are: **Bit[11], CGO[11]** Real-time (RLT) block **Bit[10], CGO[10]** Virtual residency control **Bit[9], CGO[9]** Virtual CPU communications block **Bit[8], CGO[8]** ITS communications block **Bit[7], CGO[7]** Pending table search and control **Bit[6], CGO[6]** Trace and debug **Bit[5], CGO[5]** SGI and GICR registers **Bit[4], CGO[4]** LPI cache and search **Bit[3], CGO[3]** ACE5-Lite manager interface **Bit[2], CGO[2]** ACE5-Lite subordinate interface **Bit[1], CGO[1]** SPI registers and search **Bit[0], CGO[0]** CPU communications block	0

Accessibility

GICD_FCTLR2 is accessible only by Secure accesses.

5.2.10 GICD_UTILR, Utilization Register

This register controls the utilization engine in the LPI caches. The register is not distributed and acts only on the local chip.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-10: GICD_UTILR bit assignments

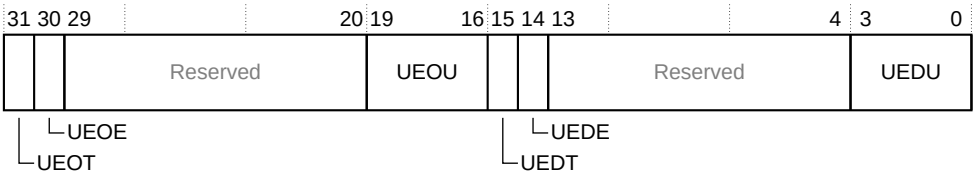


Table 5-13: GICD_UTILR bit descriptions

Out of location utilization engine settings			
Bits	**Name**	**Description**	**Type**
[31]	UEOT	Out of location utilization engine trigger. The LPI system merges LPIs of the same ID after they reach the target cache. The engine ensures optimal use of the LPI cache and it merges LPIs of the same ID that have not reached the Point-of-Serialization in the target cache. UEOE must be 1 for this bit to have any effect. No effect in configurations without LPIs.	WO
[30]	UEOE	Out of location utilization engine enable: **0** Engine is disabled **1** Enable the engine for any triggers No effect in configurations without LPIs.	RW
[29:20]	-	Reserved, RES0	-
[19:16]	UEOU	Out of location utilization engine upper threshold. Automatically trigger the engine when the LPI cache bank is UEOU/16 full.	RW

Disabled utilization engine settings			
Bits	**Name**	**Description**	**Type**
[15]	UEDT	Disabled utilization engine trigger. By default the LPI system evicts disabled LPIs as a priority when it needs space in the cache. This engine automatically evicts all disabled interrupts to improve cache performance. UEDE must be 1 for this bit to have any effect. No effect in configurations without LPIs.	WO
[14]	UEDE	Disabled utilization engine enable: **0** Engine is disabled **1** Enable the engine for any triggers No effect in configurations without LPIs.	RW
[13:4]	-	Reserved, RES0	-

Disabled utilization engine settings			
Bits	**Name**	**Description**	**Type**
[3:0]	UEDU	Disabled utilization engine upper threshold. Automatically trigger the engine when the LPI cache bank is UEDU/16 full.	RW

5.2.11 GICD_FCTLR3, Function Control Register 3

This register allows software to set some limitations on the cross-chip AXI5-Stream communications. The register is not distributed and acts only on the local chip. The GIC ignores this register for cross-chip ACE5-Lite communications, that is, when GICD_CFGID.ACE_CC == 1.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-11: GICD_FCTLR3 bit assignments

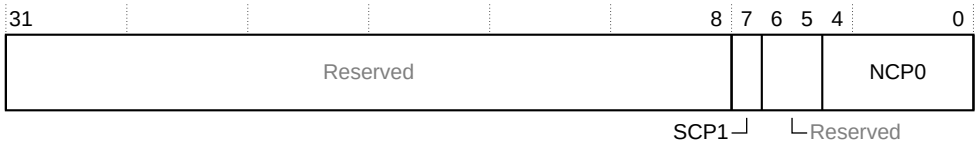


Table 5-14: GICD_FCTLR3 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RES0
[7]	SCP1	Controls whether to use separate credits for SPI and LPI commands: **0** Unified credit **1** Separate credit. This value occurs at reset. Sharing reduces the maximum number of outstanding 64-bit AXI5-Stream beats that are possible by two, if programmed in the sending and receiving chip. This bit has no effect in single-chip configurations. Any restriction limits the performance of cross-chip traffic, so if possible leave it unrestricted.
[6:5]	-	Reserved, RES0

Bits	Name	Description
[4:0]	NCP0	This field sets the maximum number of 64-bit AXI5-Stream beats between two chips. The allowable range of values for NCP0 is 6-31. The value at reset is 31. The maximum outgoing AXI5-Stream beats are 6 + NCP0 + SCP1. The maximum AXI5-Stream responses are 3 + SCP1(remote chip) + NCP0 (remote chip). This field has no effect in single-chip configurations. Any restriction limits the performance of cross-chip traffic, so if possible leave it unrestricted.

Accessibility

GICD_FCTLR3 is accessible only by Secure accesses.

5.2.12 GICD_CCCTLR, Cross-Chip Control Register

This register controls the features in the GICD that relate to an ACE5-Lite cross-chip interface. The register is not distributed and acts only on the local socket.

Configurations

This register is available in multichip configurations when GICD_CFGID.ACE_CC == 1.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-12: GICD_CCCTLR bit assignments

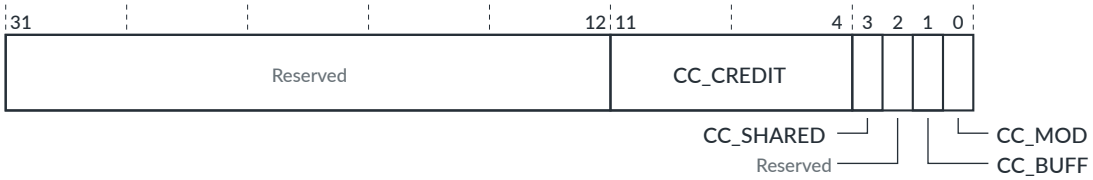


Table 5-15: GICD_CCCTLR bit descriptions

Bits	Name	Description
[31:12]	-	Reserved

Bits	Name	Description
[11:4]	CC_CREDIT	The number of credits that are available: **0xFF** 255 credits are available. **...** ... **0x02** 2 credits are available. **0x01** 1 credit is available. **0x00** The number of credits available is `ace_cc_credits`. The `ace_cc_credits` value is set during the GIC configuration stage.
[3]	CC_SHARED	Controls whether transactions are shareable: **1** All transactions use shared IDs. **0** Unordered, always use unique IDs. The cross-chip CRC scheme does not protect the bid_m signal. If the chip-chip interconnect does not support parity or perform some other check on the bid_m signal, then we strongly recommend that CC_SHARED is set to 1.
[2]	-	Reserved
[1]	CC_BUFF	Controls whether transactions are bufferable: **1** Transactions are bufferable. **0** Transactions are Non-bufferable.
[0]	CC_MOD	Controls whether transactions are bufferable: **1** Normal **0** Device

Accessibility

GICD_CCCTLR is accessible only by Secure accesses.

5.2.13 GICD_CHIPSR, Chip Status Register

This register returns the status of the chip in a multichip configuration. A single copy of this register exists on each chip in a multichip configuration.

Configurations

This register is available in all multichip configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-13: GICD_CHIPSR bit assignments

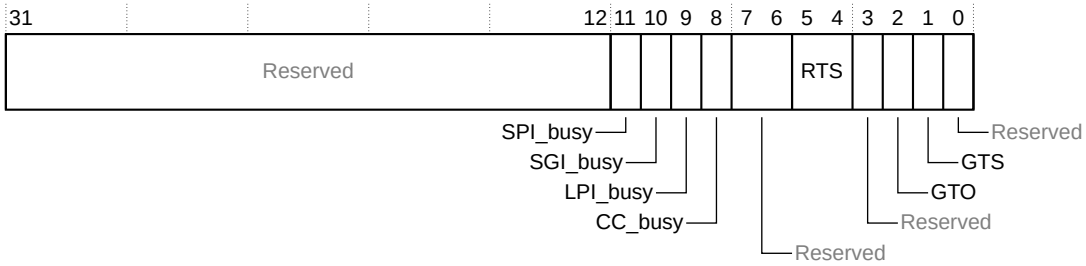


Table 5-16: GICD_CHIPSR bit descriptions

Bits	Name	Description
[31:12]	-	Reserved, RES0
[11]	SPI_busy	**0** Ongoing SPI-related cross-chip traffic. **1** No traffic.
[10]	SGI_busy	**0** Ongoing SGI-related traffic or not all cores are asleep. **1** No traffic.
[9]	LPI_busy	**0** Ongoing LPI-related traffic. **1** No traffic.
[8]	CC_busy	**0** Ongoing cross-chip traffic. **1** No traffic.
[7:6]	-	Reserved, RES0
[5:4]	RTS	Routing table status: **0b00** Disconnected. **0b01** Updating. **0b10** Consistent. **0b11** Reserved.
[3]	-	Reserved, RES0
[2]	GTO	Gating transaction ongoing: **0** No accesses. **1** Accesses ongoing.
[1]	GTS	Gating status: **0** Not gated. **1** Gated
[0]	-	Reserved, RES0

Accessibility

GICD_CHIPSR is accessible only by Secure reads.

5.2.14 GICD_DCHIPR, Default Chip Register

This register allows Secure software to access the status of a chip in a multichip system. A single copy of this register exists on each chip in a multichip configuration.

Configurations

This register is available in all multichip configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-14: GICD_DCHIPR bit assignments

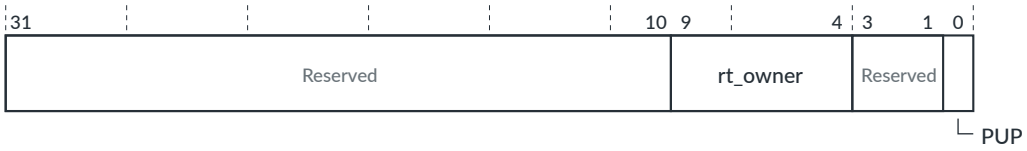


Table 5-17: GICD_DCHIPR bit assignments

Bits	Name	Description	Type
[31:10]	-	Reserved	-
[9:4]	rt_owner	Routing table owner: Value = 0-maximum chip, in the configuration.	RW
[3:1]	-	Reserved	-
[0]	PUP	Power update in progress: **0** PUP not in progress. **1** PUP in progress.	RO

Accessibility

GICD_DCHIPR is accessible only by Secure accesses.

5.2.15 GICD_CHIPR<n>, Chip Registers

Each register controls the configuration of the chip in a multichip system. This register exists on each chip in a multichip configuration and is identified by the chip number.

Configurations

This register is available in all multichip configurations.

Attributes

Width 64-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

Ignores writes if any interrupt group enable is set, that is, GICD_CTLR.EnableGrp0 == 1, or EnableGrp1NS == 1, or EnableGrp1S == 1.

Bit descriptions

Figure 5-15: GICD_CHIPR<n> bit assignments

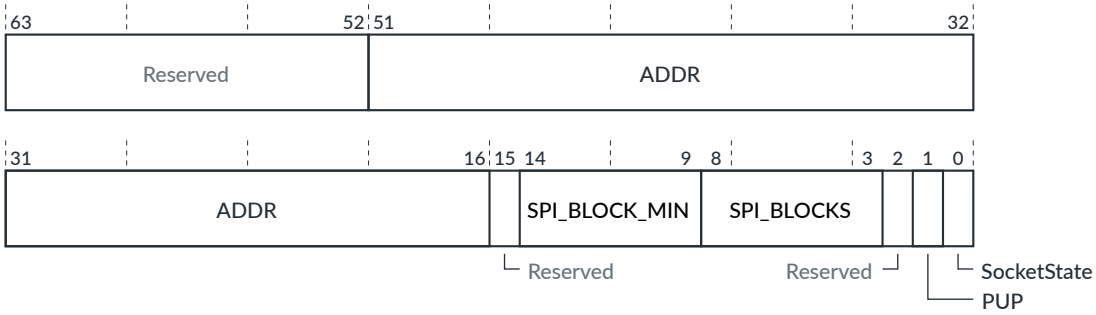


Table 5-18: GICD_CHIPR<n> bit assignments

Bits	Name	Description	Type
[63:52]	-	Reserved	-
[51:16]	ADDR	When routing messages to the remote chip, this field controls: • The value of the icdrtdest signal for an AXI5-Stream cross-chip interface. • The value of the awaddr[AXIM_ADDR_WIDTH−1:16] signal for an ACE5-Lite cross-chip interface. The `chip_addr_width` configuration parameter controls the width of this field, so the field spans from bit[16] upwards. If GICD_CFGID.ACRC == 1, then bit[16] is RES0.	RW
[15]	-	Reserved	-

Bits	Name	Description	Type
[14:9]	SPI_BLOCK_MIN	Controls the lowest number SPI block that is assigned to the chip. If the lowest SPI ID (SPI_ID) to be assigned to a chip is in the 32-991 range, calculate the value using (SPI_ID – 32)/32. If the lowest SPI_ID is in the 4096-5119 extended range, calculate the value using (SPI_ID – 4096)/32 + 30. SPIs in the 992-1023 range cannot be used as deliverable SPIs.	RW
[8:3]	SPI_BLOCKS	Controls the number of SPI blocks that are allocated to the chip. The permitted values are 0-62.	RW
[2]	-	Reserved	-
[1]	PUP	This bit returns the power update status: **0** Power update is complete. **1** Power update in progress.	RO
[0]	SocketState	This bit controls the state of the chip: **0** Chip is offline. **1** Chip is online.	RW

Accessibility

GICD_CHIPR<n> is accessible only by Secure accesses.

5.2.16 GICD_RDOFFR<n>, Redistributor Off Registers

Each register allows Secure software to remove up to 64 cores from the GIC.

Configurations

This register is available in configurations when GICD_CFGID.RDC == 1.

Attributes

Width 64-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

Software must program this register before any other GIC registers are accessed (other than reads to GICR_TYPER and the ID registers) and before the GIC receives messages from any processors. Otherwise the behavior is unpredictable.

Bit descriptions

Figure 5-16: GICD_RDOFFR<n> bit assignments

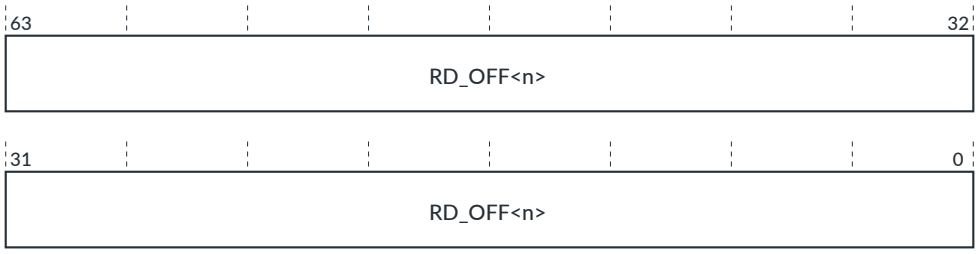


Table 5-19: GICD_RDOFFR<n> bit descriptions

Bits	Name	Description
[63:0]	RD_OFF<n>	Controls whether a core is removed from the GIC: **Bit[m] = 0** The core is not removed. **Bit[m] = 1** Removes the core that is given by 64 × <n> + m. Where <n> represents the numeric identifier of this register, that is, 0-7. The bit order in the GICD_RDOFFR register is the order that the Redistributor pages appear in the default GIC address map, as defined by the order of GCI blocks and buses within them. These values are set by the `ppi_ref` and `bus` parameters in the configuration file. When software removes cores by setting some GICD_RDOFFR bits, the GICD updates other software-visible fields to match the reduced core count. These updates include: • Moving GICR_TYPER.Last to the last Redistributor. • Moving the GICDA register page to the page above the last Redistributor. • Modifying the RAM RAS features such as scrub and error insertion, so that unused lines can never be accessed and report errors. See Limitations on page 363 for information about an MBIST limitation.

Accessibility

GICD_RDOFFR<n> is accessible only by Secure accesses.

5.2.17 GICD_VCFGBASER, vICM Final vPE CFG Attribute Register

This register returns the access attributes of the vPE Configuration table.

Configurations

This register is available in all configurations when `ppi_count` == 0, that is, there are zero GCIs.

Attributes

Width 64-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-17: GICD_VCFGBASER bit assignments

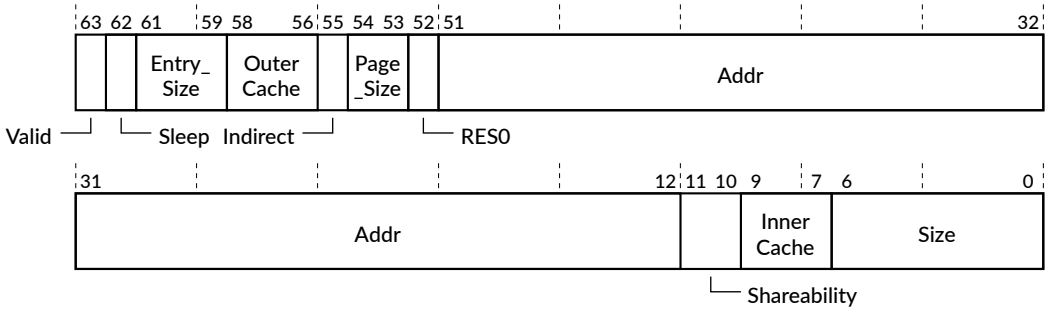


Table 5-20: GICD_VCFGBASER bit descriptions

Bits	Name	Description
[63]	Valid	Indicates whether the access attributes of the vPE Configuration table are valid: **0** The access attributes of the vPE Configuration table are not valid. **1** The access attributes of the vPE Configuration table are valid.
[62]	Sleep	Returns the value of GICD_VSLEEPR.Sleep
[61:59]	Entry_Size	Returns the value of GITS_BASER2.Entry_Size
[58:56]	OuterCache	Returns the value of GITS_BASER2.OuterCache
[55]	Indirect	Returns the value of GITS_BASER2.Indirect
[54:53]	Page_Size	Returns the value of GITS_BASER2.Page_Size
[52]	-	**RES0**
[51:12]	Addr	Returns bits[51:12] of the vPE Configuration table base address
[11:10]	Shareability	Returns the value of GITS_BASER2.Shareability
[9:7]	InnerCache	Returns the value of GITS_BASER2.InnerCache
[6:0]	Size	Returns the value of GITS_BASER2.Size

5.2.18 GICD_VSLEEPR, vICM Sleep Register

This register allows software to put the *virtual ITS Communication Module* (vICM) to sleep and drain interrupts and programming out of the GICD.

Configurations

This register is available in all configurations when `ppi_count` == 0, that is, there are zero GCIs.

Attributes

Width 32-bit

Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-18: GICD_VSLEEPR bit assignments

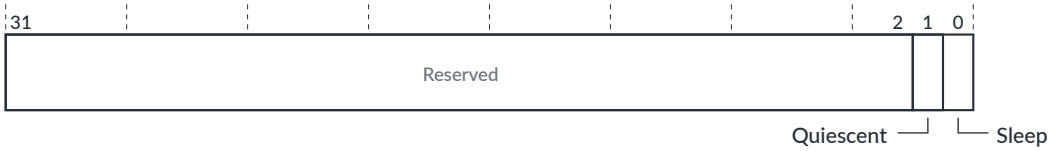


Table 5-21: GICD_VSLEEPR bit descriptions

Bits	Name	Description	Type
[31:2]	-	Reserved	-
[1]	Quiescent	Indicates whether the vICM is active: **0** vICM is awake **1** vICM is asleep	RO
[0]	Sleep	Controls whether the vICM is asleep: **0** Abandon sleep **1** Put vICM to sleep and drain interrupts and programming out of the GICD.	RW

5.2.19 GICD_ICLARn, Interrupt Class Registers

These registers control whether a 1 of N SPI can target a core that is assigned to class 0 or class 1 group. Each register controls 16 SPIs and the GIC-720AE has 60 registers, GICD_ICLAR2-GICD_ICLAR61.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 60 registers to support the first 960 SPIs. If you configure the GIC-720AE to use fewer than 960 SPIs, then it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI. See also GICD_ICLARnE.

Bit descriptions

Figure 5-19: GICD_ICLARn bit assignments

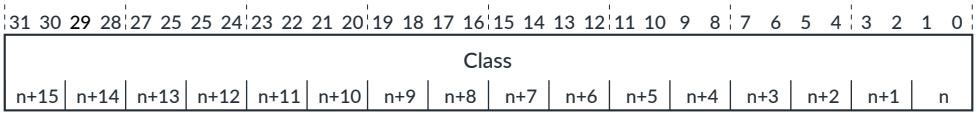


Table 5-22: GICD_ICLARn bit descriptions

Bits	Name	Description
[31:0] Bits[2x+1:2x], for x = 0 to 15	Class<x>	Controls whether the 1 of N SPI can target a core, depending on the class group that the core is assigned to: **0b00** The SPI can target a core that is assigned to class 0 or class 1. **0b01** The SPI can target a core that is assigned to class 1. **0b10** The SPI can target a core that is assigned to class 0. **0b11** The SPI cannot target a core that is assigned to class 0 or class 1. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ICLARn, that is, SPI = 16×n + bit[number]/2.

Accessibility

GICD_ICLARn is accessible only when the corresponding GICD_IROUTERn.Interrupt_Routing_Mode == 1.

5.2.20 GICD_ICERRRn, Interrupt Clear Error Registers

These registers can clear the error status of an SPI or return the error status of an SPI. Each register monitors 32 SPIs and the GIC-720AE has 30 registers, GICD_ICERRR1-GICD_ICERRR30.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 30 registers to support 960 SPIs. If you configure the GIC-720AE to use fewer than 960 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-20: GICD_ICERRRn bit assignments

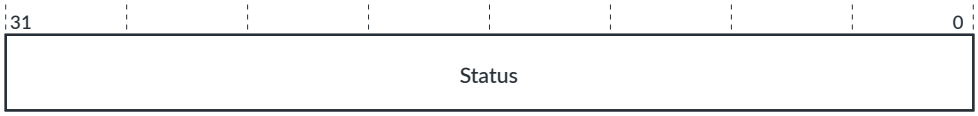


Table 5-23: GICD_ICERRRn bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether an SPI is in an error state: **0** If read, the SPI is not in an error state and programming is valid. Writing 0 has no effect. **1** If read, the SPI is in an error state and programming is not valid. Writing 1 clears the error. Non-secure software can access this register, only if Secure software has previously used the GICD_ICGERRn or GICD_ICGERRnE to clear the group information, and it has reprogrammed the group. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ICERRRn, that is, SPI = 32×n + bit[number].

5.2.21 GICD_ICGERRn, Interrupt Clear Group Error registers

These registers can clear the error status of the GICD_IGROUPRn, GICD_IGRPMODRn, and GICD_NSACRn registers of an SPI or return the error status of an SPI. Each register monitors 32 SPIs and the GIC-720AE has 30 registers, GICD_ICGERR1-GICD_ICGERR30.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 30 registers to support 960 SPIs. If you configure the GIC-720AE to use fewer than 960 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-21: GICD_ICGERRn bit assignments

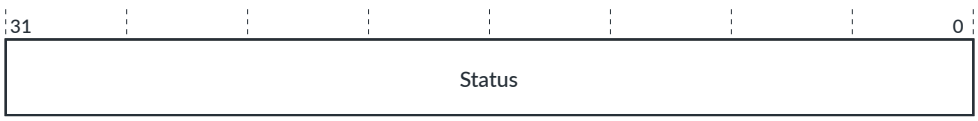


Table 5-24: GICD_ICGERRn bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether an SPI is in an error state: **0** If read, the SPI is not in an error state and programming is valid. Writing 0 has no effect. **1** If read, the SPI is in an error state and programming is not valid. Writing 1 clears the error group information. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ICGERRn, that is, SPI = 32×n + bit[number].

Accessibility

GICD_ICGERRn is accessible only by Secure accesses.

5.2.22 GICD_ISERRRn, Interrupt Set Error Registers

These registers can set the error status of an SPI or return the error status of an SPI. Each register monitors 32 SPIs and the GIC-720AE has 30 registers, GICD_ISERRR1-GICD_ISERRR30. Software can use these registers to test the operation of its interrupt error clear function.

When multi view support is enabled, the error that is set depends on the view:

- Accesses to view 0 set the View Error, Group Error, and Error bits.
- Accesses to view 1, 2, or 3 set the Group Error and Error bits.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 30 registers to support 960 SPIs. If you configure the GIC-720AE to use fewer than 960 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-22: GICD_ISERRRn bit assignments

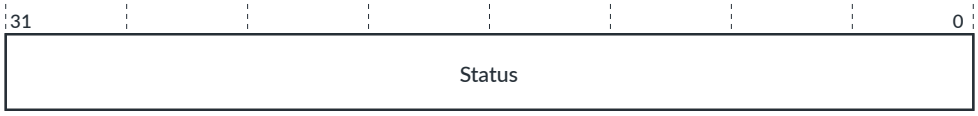


Table 5-25: GICD_ISERRRn bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether an SPI is in an error state: **0** If read, the SPI is not in an error state and programming is valid. Writing 0 has no effect. **1** If read, the SPI is in an error state and programming is not valid. Writing 1 sets the error and contains the SPI. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ISERRRn, that is, SPI = 32×n + bit[number].

Accessibility

GICD_ISERRRn is accessible only by Secure accesses.

5.2.23 GICD_ERRINSRn, Error Insertion Registers

This register can insert errors into the internal RAMs. You can use this register to test your error recovery software.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

See 4.6.1 RAM error simulation on page 65 for which RAM corresponds to the register suffix identifier n.

The bit assignments within this register depend on whether a write access or read access occurs.

The following table shows the bit assignments for write accesses.

Table 5-26: GICD_ERRINSRn bit assignments for writes

Bits	Name	Description
[63]	Valid	Set to 1, to start the error injection process. The GIC sets this bit to 0 when it completes the process.
[62:61]	-	**RES0**
[60]	DisableWriteCheck	Controls whether to include an encoding check: **0** Include an encoder check. **1** Disable an encoder check.

Bits	Name	Description
[59:48]	-	**RES0**
[47:32]	ADDR	Address
[31]	ERRINS2VALID	Controls whether the second error is valid: **0** The ERRINS2LOC field is not valid. **1** The ERRINS2LOC field is valid.
[30:25]	-	**RES0**
[24:16]	ERRINS2LOC	Sets the bit location of the second error.
[15]	ERRINS1VALID	Controls whether the first error is valid: **0** The ERRINS1LOC field is not valid. **1** The ERRINS1LOC field is valid.
[14:9]	-	**RES0**
[8:0]	ERRINS1LOC	Sets the bit location of the first error.

The following table shows the bit assignments for read accesses.

Table 5-27: GICD_ERRINSRn bit assignments for reads

Bits	Name	Description
[63]	Valid	Indicates if the error injection process is complete: **0** Error injection process is complete. **1** Error injection process is in progress.
[62:61]	Status	Indicates if the error injection process was successful, and is valid only when Valid == 0: **0b00** The GIC performed the error injection process. **0b01** An out-of-range error occurred. To fix this error, check that the RAM ID and the error locations are correct. **0b10** A coincident error occurred. **0b11** An encoder or decoder mismatch occurred.
[60]	RAM_Present	Indicates whether a RAM with ECC is present: **1** RAM with ECC is present.
[59:48]	-	**RES0**
[47:32]	RAM_MAX	Returns the maximum address of the RAM.
[31:9]	-	**RES0**
[8:0]	RAM WIDTH	Returns the highest maximum bit width of the RAM. For example, a value of 15 indicates a 16-bit wide RAM.

Accessibility

If GICD_SAC.GICTNS == 0, then GICD_ERRINSRn is accessible only by Secure accesses.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, GICD_ERRINSRn is accessible only from view 0.

5.2.24 GICD_ICLARnE, Interrupt Class Registers Extended

These registers control whether a 1 of N SPI can target a core that is assigned to class 0 or class 1 group. Each register controls 16 SPIs and the GIC-720AE has 64 registers, GICD_ICLAR0E-GICD_ICLAR63E.

Configurations

This register is available in all configurations with > 960 SPIs.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 64 registers to support the extended SPIs, 961-1984. If you configure the GIC-720AE to use fewer than 1984 SPIs, then it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-23: GICD_ICLARnE bit assignments

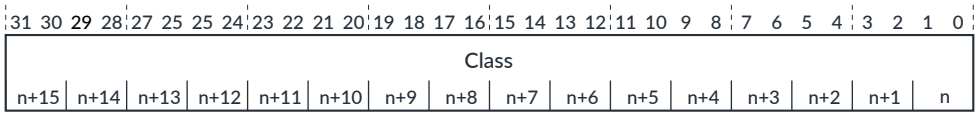


Table 5-28: GICD_ICLARnE bit descriptions

Bits	Name	Description
[31:0] Bits[2x+1:2x], for x = 0 to 15	Class<x>	Controls whether the 1 of N SPI can target a core, depending on the class group that the core is assigned to: **0b00** The SPI can target a core that is assigned to class 0 or class 1 **0b01** The SPI can target a core that is assigned to class 1 **0b10** The SPI can target a core that is assigned to class 0 **0b11** The SPI cannot target a core that is assigned to class 0 or class 1 The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ICLARnE, that is, SPI = 960 + 16×n + bit[number]/2.

5.2.25 GICD_ICERRRnE, Interrupt Clear Error Registers Extended

These registers can clear the error status of an SPI in the extended SPI range, or return the error status of an SPI. Each register monitors 32 SPIs and the GIC-720AE has up to 32 registers, GICD_ICERRR0E-GICD_ICERRR31E.

Configurations

This register is available in all configurations with > 960 SPIs.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 32 registers to support the extended SPIs, 961-1984. If you configure the GIC-720AE to use fewer than 1984 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-24: GICD_ICERRRnE bit assignments

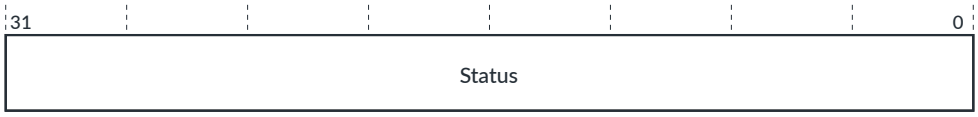


Table 5-29: GICD_ICERRRnE bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether an SPI is in an error state: **0** If read, the SPI is not in an error state and programming is valid. Writing 0 has no effect. **1** If read, the SPI is in an error state and programming is not valid. Writing 1 clears the error. Non-secure software can access this register, only if Secure software has previously used the 5.2.21 GICD_ICGERRn, Interrupt Clear Group Error registers on page 167 or 5.2.26 GICD_ICGERRnE, Interrupt Clear Group Error registers Extended on page 172 to clear the group information, and it has reprogrammed the group. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ICERRRnE, that is, SPI = 960 + 32×n + bit[number].

5.2.26 GICD_ICGERRnE, Interrupt Clear Group Error registers Extended

These registers can clear the error status of the GICD_IGROUPRnE, GICD_IGRPMODRnE, and GICD_NSACRnE registers of an SPI, or it returns the error status of an SPI. Each register monitors 32 SPIs and the GIC-720AE has up to 32 registers, GICD_ICGERR0E-GICD_ICGERR31E.

Configurations

This register is available in all configurations with > 960 SPIs.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 32 registers to support the extended SPIs, 961-1984. If you configure the GIC-720AE to use fewer than 1984 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-25: GICD_ICGERRnE bit assignments

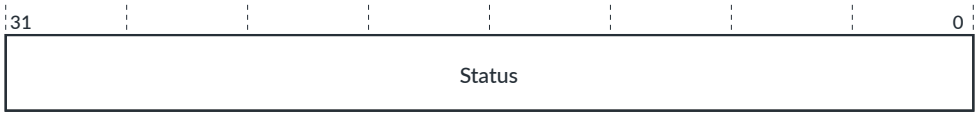


Table 5-30: GICD_ICGERRnE bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether an SPI is in an error state: **0** If read, the SPI is not in an error state and programming is valid. Writing 0 has no effect. **1** If read, the SPI is in an error state and programming is not valid. Writing 1 clears the error group information. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ICGERRnE, that is, SPI = 960 + 32×n + bit[number].

Accessibility

GICD_ICGERRnE is accessible only by Secure accesses.

5.2.27 GICD_ISERRRnE, Interrupt Set Error Registers Extended

These registers can set the error status of an SPI in the extended SPI range, or return the error status of an SPI. Each register monitors 32 SPIs and the GIC-720AE has up to 32 extended

registers, GICD_ISERRR0E-GICD_ISERRR31E. Software can use these registers to test the operation of its interrupt error clear function.

When multi view support is enabled, the error that is set depends on the view:

- Accesses to view 0 set the View Error, Group Error, and Error bits.
- Accesses to view 1, 2, or 3 set the Group Error and Error bits.

Configurations

This register is available in all configurations with > 960 SPIs.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 32 registers to support the extended SPIs, 961-1984. If you configure the GIC-720AE to use fewer than 1984 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-26: GICD_ISERRRnE bit assignments

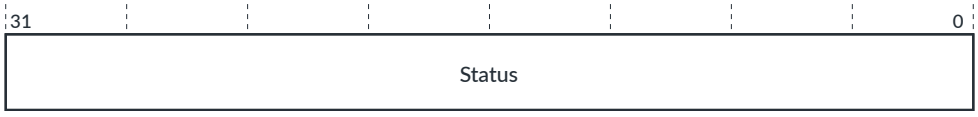


Table 5-31: GICD_ISERRRnE bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether an SPI is in an error state: **0** If read, the SPI is not in an error state and programming is valid. Writing 0 has no effect. **1** If read, the SPI is in an error state and programming is not valid. Writing 1 sets the error and contains the SPI. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ISERRRnE, that is, SPI = 960 + 32×n + bit[number].

Accessibility

GICD_ISERRRnE is accessible only by Secure accesses.

5.2.28 GICD_CFGID, Configuration ID Register

This register contains information that enables test software to determine if the GIC-720AE system is compatible.

Configurations

This register is available in all configurations.

Attributes

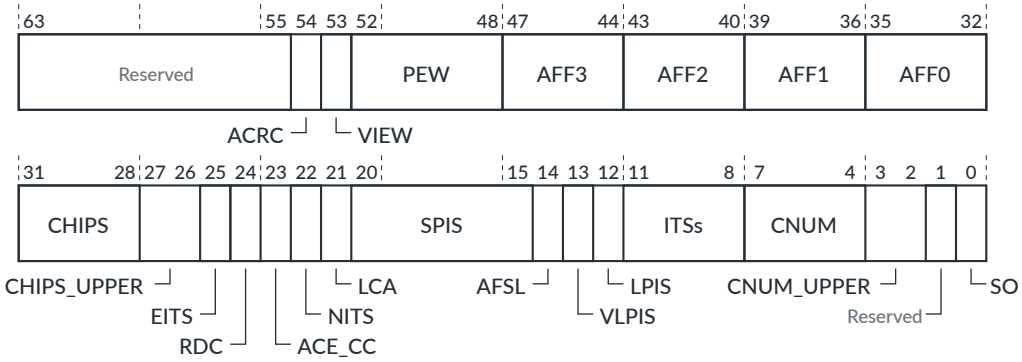
Width 64-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-27: GICD_CFGID bit assignments



In the following table, the View column is applicable only for GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1.

Table 5-32: GICD_CFGID bit assignments

Bits	Name	Description	View
[63:55]	-	Reserved, returns zero	-
[54]	ACRC	Indicates whether the GIC has ACE5-Lite cross-chip interface that includes CRC protection: **0** The GIC configuration does not have a ACE5-Lite cross-chip interface with cross-chip stream CRC protection. **1** The GIC configuration has a ACE5-Lite cross-chip interface with cross-chip CRC protection. This value occurs when all the following are true: • The ACE_CC bit is set to 1. • The `cc_stream_protection_type` configuration parameter is set to 2 or 3.	0

Bits	Name	Description	View
[53]	VIEW	Indicates whether the GIC supports multi view: **0** The GIC does not support multi view. **1** The GIC supports multi view. This value occurs when the `multi_view_support` configuration parameter is set to 1.	0, 1, 2, 3
[52:48]	PEW	Width of lower part of on-chip core number field, ceil[$\log_2$(max_pe_on_chip)]. `max_pe_on_chip` is a configuration option that is set during system integration, which defines the maximum number of cores on a single chip in the system. See 4.13.3 LPI multichip operation on page 87 for more information.	0, 1, 2, 3
[47:44]	AFF3	Returns the Affinity3 bits.	0, 1, 2, 3
[43:40]	AFF2	Returns the Affinity2 bits.	0, 1, 2, 3
[39:36]	AFF1	Returns the Affinity1 bits.	0, 1, 2, 3
[35:32]	AFF0	Returns the Affinity0 bits.	0, 1, 2, 3
[31:28]	CHIPS	Returns the number of supported chips – 1[3:0].	0, 1, 2, 3
[27:26]	CHIPS_UPPER	Returns the number of supported chips – 1[5:4].	0, 1, 2, 3
[25]	EITS	Returns 1 when the GIC supports more than 16 ITSs.	0, 1. Returns zero for views 2 and 3.
[24]	RDC	Redistributor collapse. A Secure read indicates whether the GIC enables Secure software to program the core numbering: **0** Secure software cannot program the core numbering. **1** Secure software can program the core numbering by programming GICD_RDOFFRn and GICR_MPIDR. This bit is set to 1 when `prog_mpidr == prog`. The `prog_mpidr` parameter is set during configuration of the GIC. See A.1 Removing cores from a preconfigured GIC on page 361 for more information.	0. Returns zero for views 1, 2 and 3.
[23]	ACE_CC	Indicates the AMBA® protocol that the cross-chip interface uses: **0** The cross-chip interface uses the AXI5-Stream protocol. **1** The cross-chip interface uses the ACE5-Lite protocol. The cross-chip interface is not present when CHIPS == CHIPS_UPPER == 0.	0. Returns zero for views 1, 2 and 3.
[22]	NITS	No ITS present. Indicates whether a local ITS is present: **0** The chip contains a local ITS. **1** The chip has no local ITS. Returns zero if LPIS == 0 (no LPI support).	0, 1. Returns zero for views 2 and 3.
[21]	LCA	Local chip addressing: **0** All chips use the same addressing scheme to communicate with another chip. **1** Each chip can use its own local addressing scheme when it communicates with another chip. See Local cross-chip addressing on page 59 for more information.	0. Returns zero for views 1, 2 and 3.
[20:15]	SPIS	Number of SPI blocks supported.	0, 1, 2, 3
[14]	AFSL	Chip affinity selection level.	0, 1, 2, 3

Bits	Name	Description	View
[13]	VLPIS	GICv4.1 supported	0, 1. Returns zero for views 2 and 3.
[12]	LPIS	LPI supported	0, 1. Returns zero for views 2 and 3.
[11:8]	ITSs	The number of supported ITSs minus 1. When: • EITS == 0, then the ITSs field represents 0-15. • EITS == 1, then the ITSs field represents 16-31. Returns zero if LPIS == 0 (no LPI support).	0, 1. Returns zero for views 2 and 3.
[7:4]	CNUM	Chip number[3:0]	0, 1, 2, 3
[3:2]	CNUM_UPPER	Chip number[5:4]	0, 1, 2, 3
[1]	-	Reserved, returns zero	-
[0]	SO	Socket online status: **0** Chip is offline. **1** Chip is online.	0, 1, 2, 3

Accessibility

The RDC bit is accessible only by Secure accesses.

5.2.29 GICD_ICVERRRn, Interrupt Clear View Error Registers

These registers can clear the view error status of an SPI or return the error status of an SPI. Each register monitors 32 SPIs and the GIC-720AE has 30 registers, GICD_ICVERRR1-GICD_ICVERRR30.

Configurations

These registers are available in configurations that support multi view, that is, when GICD_CFGID.VIEW == 1.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 30 registers to support 960 SPIs. If you configure the GIC-720AE to use fewer than 960 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-28: GICD_ICVERRRn bit assignments

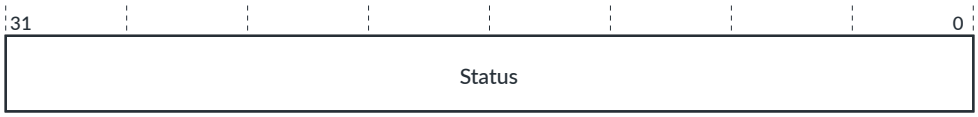


Table 5-33: GICD_ICVERRRn bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether an SPI is in an error state: **0** If read, the SPI is not in an error state and programming is valid. Writing 0 has no effect. **1** If read, the SPI is in an error state and programming is not valid. Writing 1 clears the view error. Non-secure software can access this register, only if Secure software has previously used the GICD_ICGERRn or GICD_ICGERRnE to clear the group information, and it has reprogrammed the group. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ICVERRRn, that is, SPI = 32×n + bit[number].

Accessibility

GICD_ICVERRRn is accessible only for view 0.

5.2.30 GICD_ICVERRRnE, Interrupt Clear View Error Registers Extended

These registers can clear the view error status of an SPI in the extended SPI range, or return the error status of an SPI. Each register monitors 32 SPIs and the GIC-720AE has up to 32 registers, GICD_ICVERRR0E-GICD_ICVERRR31E.

Configurations

These registers are available only in configurations that support all of the following features:

- > 960 SPIs.
- Multi view, that is, when GICD_CFGID.VIEW == 1.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 32 registers to support the extended SPIs, 961-1984. If you configure the GIC-720AE to use fewer than 1984 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-29: GICD_ICVERRRnE bit assignments

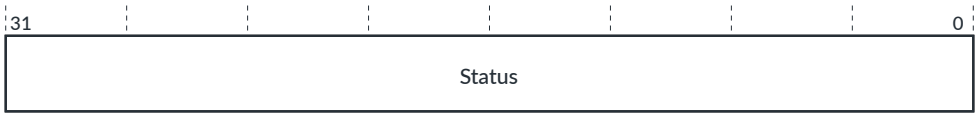


Table 5-34: GICD_ICVERRRnE bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether an SPI is in an error state: **0** If read, the SPI is not in an error state and programming is valid. Writing 0 has no effect. **1** If read, the SPI is in an error state and programming is not valid. Writing 1 clears the view error. Non-secure software can access this register, only if Secure software has previously used the 5.2.21 GICD_ICGERRn, Interrupt Clear Group Error registers on page 167 or 5.2.26 GICD_ICGERRnE, Interrupt Clear Group Error registers Extended on page 172 to clear the group information, and it has reprogrammed the group. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_ICVERRRnE, that is, SPI = 960 + 32×n + bit[number].

Accessibility

GICD_ICVERRRnE is accessible only for view 0.

5.2.31 GICD_IVIEWRn, Interrupt View Registers

These registers control whether an SPI is assigned to view 0, view 1, view 2, or view 3. Each register controls 16 SPIs and the GIC-720AE has 60 registers, GICD_IVIEWR2-GICD_IVIEWR61.

Configurations

This register is available in configurations that support multi view, that is, when GICD_CFGID.VIEW == 1.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 60 registers to support the first 960 SPIs. If you configure the GIC-720AE to use fewer than 960 SPIs, then it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI. See also GICD_IVIEWRnE.

Bit descriptions

Figure 5-30: GICD_IVIEWRn bit assignments

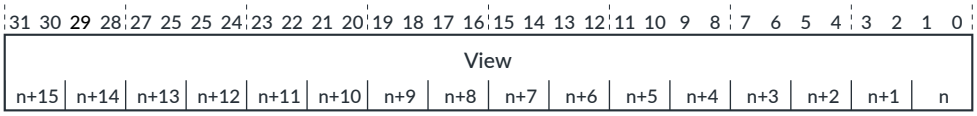


Table 5-35: GICD_IVIEWRn bit descriptions

Bits	Name	Description
[31:0] Bits[2x+1:2x], for x = 0 to 15	View<x>	Controls the allocation of SPIs to a view: **0b00** The SPI is assigned to view 0. **0b01** The SPI is assigned to view 1. **0b10** The SPI is assigned to view 2. **0b11** The SPI is assigned to view 3. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_IVIEWRn, that is, SPI = 16×n + bit[number]/2.

Accessibility

GICD_IVIEWRn is accessible only for view 0.

5.2.32 GICD_IVIEWRnE, Interrupt View Registers Extended

These registers control whether an SPI is assigned to view 0, view 1, view 2, or view 3. Each register controls 16 SPIs and the GIC-720AE has 64 registers, GICD_IVIEWR0E-GICD_IVIEWR63E.

Configurations

These registers are available only in configurations that support all of the following features:

- > 960 SPIs.
- Multi view, that is, when GICD_CFGID.VIEW == 1.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 64 registers to support the extended SPIs, 961-1984. If you configure the GIC-720AE to use fewer than 1984 SPIs, then it reduces the number of registers accordingly. For locations where interrupts are not implemented, the register is RAZ/WI.

Bit descriptions

Figure 5-31: GICD_IVIEWRnE bit assignments

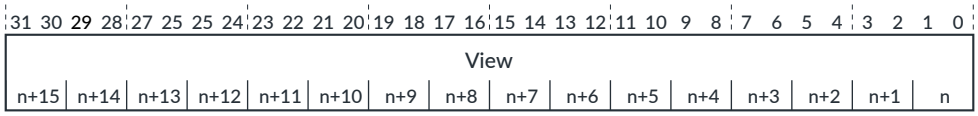


Table 5-36: GICD_IVIEWRnE bit descriptions

Bits	Name	Description
[31:0] Bits[2x+1:2x], for x = 0 to 15	View<x>	Controls the allocation of SPIs to a view: **0b00** The SPI is assigned to view 0. **0b01** The SPI is assigned to view 1. **0b10** The SPI is assigned to view 2. **0b11** The SPI is assigned to view 3. The SPI that a bit refers to, depends on its bit position and the base address offset of the GICD_IVIEWRnE, that is, SPI = 960 + 16×n + bit[number]/2.

Accessibility

GICD_IVIEWRnE is accessible only for view 0.

5.2.33 GICD_PIDR4, Peripheral ID4 register

This register returns byte[4] of the peripheral ID. The GICD_PIDR4 register is part of the set of Distributor peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-32: GICD_PIDR4 bit assignments

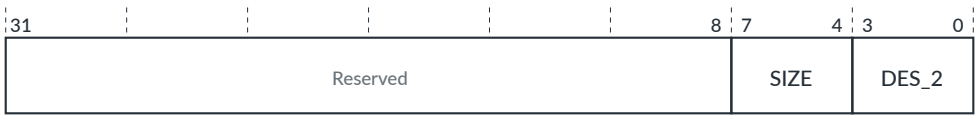


Table 5-37: GICD_PIDR4 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:4]	SIZE	Returns `0x4`, which indicates that the Distributor occupies 64KB of memory, (2^{SIZE} × 4KB).
[3:0]	DES_2	Returns `0x4`, which represents bits[10:7] of the JEDEC JEP106 identification code. Together, GICD_PIDR1.DES_0, GICD_PIDR2.DES_1, and DES_2 identify the component designer.

5.2.34 GICD_PIDR3, Peripheral ID3 register

This register returns byte[3] of the peripheral ID. The GICD_PIDR3 register is part of the set of Distributor peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-33: GICD_PIDR3 bit assignments

Table 5-38: GICD_PIDR3 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:4]	REVAND	Indicates minor errata fixes specific to the revision of the component being used, for example metal fixes after implementation. `0x0` indicates that there are no errata fixes to this component. `0x0`.
[3:0]	CMOD	Customer modified. Indicates whether the customer has modified the behavior of the component. Usually, this field is `0x0`. Customers change this value when they make authorized modifications to this component. `0x0`.

5.2.35 GICD_PIDR2, Peripheral ID2 register

This register returns byte[2] of the peripheral ID. The GICD_PIDR2 register is part of the set of Distributor peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-34: GICD_PIDR2 bit assignments

Table 5-39: GICD_PIDR2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:4]	ArchRev	Identifies the version of the GIC architecture with which the Distributor complies: **0x3** GICv3 **0x4** GICv4
[3]	JEDEC	Indicates that a JEDEC-assigned JEP106 identity code is used.
[2:0]	DES_1	Bits[6:4] of the JEP106 identity code. Bits[3:0] of the JEP106 identity code are assigned to GICD_PIDR1[7:4].

5.2.36 GICD_PIDR1, Peripheral ID1 register

This register returns byte[1] of the peripheral ID. The GICD_PIDR1 register is part of the set of Distributor peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-35: GICD_PIDR1 bit assignments

Table 5-40: GICD_PIDR1 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:4]	DES_0	Returns `0xB`, which represents bits[3:0] of the JEDEC JEP106 identification code. Together, DES_0, GICD_PIDR2.DES_1, and GICD_PIDR4.DES_2 identify the component designer.
[3:0]	PART_1	Returns `0x4`, which represents bits[11:8] of the 12-bit part number of the Distributor. Together, GICD_PIDR0.PART_0 and PART_1 field values indicate the part number of the Distributor.

5.2.37 GICD_PIDR0, Peripheral ID0 register

This register returns byte[0] of the peripheral ID. The GICD_PIDR0 register is part of the set of Distributor peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 135 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-36: GICD_PIDR0 bit assignments

Table 5-41: GICD_PIDR0 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:0]	PART_0	Returns `0x92`, which represents bits[7:0] of the 12-bit part number of the Distributor. Together, PART_0 and GICD_PIDR1.PART_1 field values indicate the part number of the Distributor.

5.3 Distributor registers (GICM) for message-based SPIs summary

The functions for the GIC-720AE message-based SPIs are controlled through the Distributor registers identified with the prefix GICM.

The following table lists the message-based SPI registers in base offset order and provides a reference to the register description that is described in either this document or the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*. The WO registers allow 16-bit accesses.

Table 5-42: Distributor registers (GICM) for message-based SPIs summary

Offset	Name	Type	Reset	Width	Description	Architecture defined?
`0x0000-0x0004`	-	-	-	-	Reserved	-
`0x0008`	GICM_TYPER	RO	Configuration dependent	64	Message-based Type Register	Yes
`0x0010-0x003C`	-	-	-	-	Reserved	-
`0x0040`	GICM_SETSPI_NSR	WO	-	32	Message-based Non-secure SPI Set Register	Yes
`0x0044`	-	-	-	-	Reserved	-
`0x0048`	GICM_CLRSPI_NSR	WO	-	32	Message-based Non-secure SPI Clear Register	Yes
`0x004C`	-	-	-	-	Reserved	-
`0x0050`	GICM_SETSPI_SR	WO	-	32	Message-based Secure SPI Set Register. Only present when Security support is included, otherwise Reserved.	Yes
`0x0054`	-	-	-	-	Reserved	-
`0x0058`	GICM_CLRSPI_SR	WO	-	32	Message-based Secure SPI Clear Register. Only present when Security support is included, otherwise Reserved.	Yes

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x005C-0x0FC8	-	-	-	-	Reserved	-
0x0FCC	GICM_IIDR	RO	0x070nn43B The nn value depends on the rxpy identifier.	32	Message-based Distributor Implementer Identification Register	Yes
0x0FD0-0xFFCC	-	-	-	-	Reserved	-
0xFFD0	GICM_PIDR4	RO	0x44	32	Peripheral ID 4 register	No
0xFFD4	GICM_PIDR5	RO	0x00	32	Peripheral ID 5 register	No
0xFFD8	GICM_PIDR6	RO	0x00	32	Peripheral ID 6 register	No
0xFFDC	GICM_PIDR7	RO	0x00	32	Peripheral ID 7 register	No
0xFFE0	GICM_PIDR0	RO	0x97	32	Peripheral ID 0 register	No
0xFFE4	GICM_PIDR1	RO	0xB4	32	Peripheral ID 1 register	No
0xFFE8	GICM_PIDR2	RO	0x3B	32	Peripheral ID 2 register	No
0xFFEC	GICM_PIDR3	RO	0x00	32	Peripheral ID 3 register	No
0xFFF0	GICM_CIDR0	RO	0x0D	32	Component ID 0 register	No
0xFFF4	GICM_CIDR1	RO	0xF0	32	Component ID 1 register	No
0xFFF8	GICM_CIDR2	RO	0x05	32	Component ID 2 register	No
0xFFFC	GICM_CIDR3	RO	0xB1	32	Component ID 3 register	No

5.3.1 GICM_TYPER, Message-based Type Register

This register returns information about the number of SPIs that are assigned to the frame.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.3 Distributor registers (GICM) for message-based SPIs summary on page 185 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-37: GICM_TYPER bit assignments

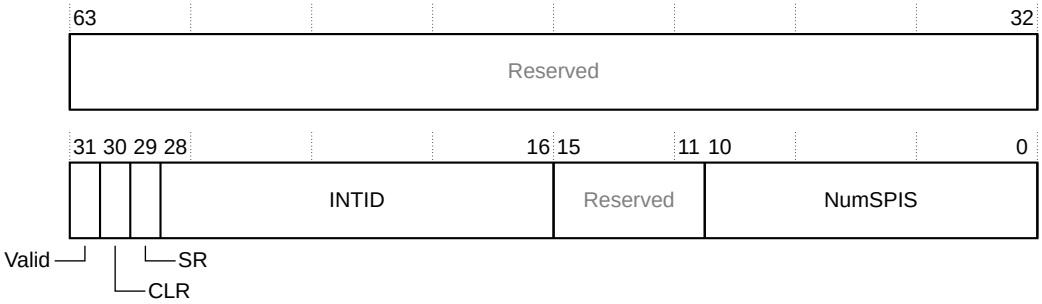


Table 5-43: GICM_TYPER bit descriptions

Bits	Name	Description
[63:32]	-	Reserved, RES0
[31]	Valid	Returns 1 to indicate that the register reports information about the capabilities of the frame.
[30]	CLR	Returns 1 to indicate that the GICM_CLRSPI registers are present.
[29]	SR	Indicates whether the GICM_CLRSPI_SR and GICM_SETSPI_SR registers are present: **0** GICM_CLRSPI_SR and GICM_SETSPI_SR registers are not present because GICD_CTLR.DS == 1. **1** GICM_CLRSPI_SR and GICM_SETSPI_SR registers are present.
[28:16]	INTID	The INTID of the lowest or first SPI that is assigned to the frame.
[15:11]	-	Reserved, RES0
[10:0]	NumSPIS	Returns the number of SPIs that are assigned to the frame. If the software is written for GICv2m, then we recommend setting GICT_ERR<n>CTLR.DIS_SPI_OOR to `0b10` or `0b01`. These values ensure that errors are not generated if software attempts to use the unimplemented SPI block with INTIDs 992-1023.

5.3.2 GICM_IIDR, Message-based Distributor Implementer Identification Register

This register provides information about the implementer and revision of the message-based Distributor page.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.3 Distributor registers (GICM) for message-based SPIs summary on page 185 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-38: GICM_IIDR bit assignments

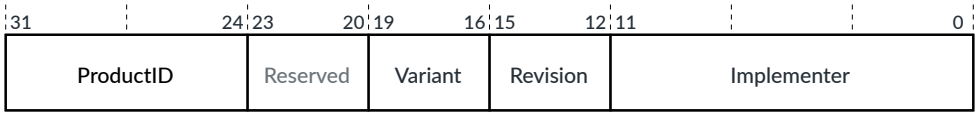


Table 5-44: GICM_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **0x07** GIC-720AE
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: **0x0** r0 **0x1** r1 **0x2** r2
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: **0x0** p0 **0x1** p1
[11:0]	Implementer	Identifies the implementer: **0x43B** Arm

5.4 Redistributor registers for control and physical LPIs summary

The functions for the GIC-720AE physical LPIs are controlled through the Redistributor registers identified with the prefix GICR. These registers start from the base address of the Redistributor.

For more information about LPIs, see the *Locality-Specific Peripheral Interrupts, Arm® Generic Interrupt Controller v3 and v4*.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, these GICR registers are accessible for view 0 and when the view matches the PE. See Multi view access to the GICR registers on page 190 for information about the views that a register is accessible in.

For descriptions of registers that are not specific to the GIC-720AE, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Table 5-45: Redistributor registers for control and physical LPIs summary

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0000	GICR_CTLR	RW	Configuration dependent	32	Redistributor Control Register	Yes
0x0004	GICR_IIDR	RO	0x070nn43B The nn value depends on the r*x*p*y* identifier.	32	Redistributor Implementation Identification Register	Yes
0x0008	GICR_TYPER	RO	Configuration dependent	64	Redistributor Type Register	Yes
0x0010	-	-	-	-	Reserved	-
0x0014	GICR_WAKER	RW	0x6	32	Power Management Control Register	Some bits are architecture defined and the others are microarchitecture defined.
0x0018	GICR_MPAMIDR	RO	0x000101FF	32	Report maximum PARTID and PMG Register	Yes
0x001C	GICR_PARTIDR	RW	0x0	32	Set PARTID and PMG Register	Yes
0x0020	GICR_FCTLR	RW	0x00000200 if the GCI supports real-time interrupts, otherwise 0x0.	32	Function Control Register	No
0x0024	GICR_PWRR	RW	Configuration dependent	32	Power Register	No
0x0028	GICR_CLASSR	RW	0x0	32	Class Register	No
0x0070	GICR_PROPBASER	RW	Configuration dependent	64	Redistributor Properties Base Address Register. Only present when ITS and LPI support support is included, otherwise Reserved.	Yes
0x0078	GICR_PENDBASER	RW	Configuration dependent	64	Redistributor LPI Pending Table Base Address Register. Only present when ITS and LPI support support is included, otherwise Reserved.	Yes
0x0080-0x009C	-	-	-	-	Reserved	-
0x00A0	GICR_INVLPIR	WO	-	64	-	Yes
0x00A8-0x00AC	-	-	-	-	Reserved	-
0x00B0	GICR_INVALLR	WO	-	64	-	Yes
0x00B8-0x00BC	-	-	-	-	Reserved	-
0x00C0	GICR_SYNCR	RO	0x0	32	-	Yes
0x00C4-0x00FC	-	-	-	-	Reserved	-
0x0100	GICR_MPIDR	WO	-	32	MPIDR Register. Present only when GICD_CFGID.RDC == 1.	No
0x0104-0xFFCC		-	-	-	Reserved	-

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0xFFD0	GICR_PIDR4	RO	0x44	32	Peripheral ID 4 Register	No
0xFFD4	GICR_PIDR5	RO	0x00	32	Peripheral ID 5 Register	No
0xFFD8	GICR_PIDR6	RO	0x00	32	Peripheral ID 6 Register	No
0xFFDC	GICR_PIDR7	RO	0x00	32	Peripheral ID 7 Register	No
0xFFE0	GICR_PIDR0	RO	0x93	32	Peripheral ID 0 Register	No
0xFFE4	GICR_PIDR1	RO	0xB4	32	Peripheral ID 1 Register	No
0xFFE8	GICR_PIDR2	RO	Configuration dependent	32	Peripheral ID 2 Register	No
0xFFEC	GICR_PIDR3	RO	0x00	32	Peripheral ID 3 Register	No
0xFFF0	GICR_CIDR0	RO	0x0D	32	Component ID 0 Register	No
0xFFF4	GICR_CIDR1	RO	0xF0	32	Component ID 1 Register	No
0xFFF8	GICR_CIDR2	RO	0x05	32	Component ID 2 Register	No
0xFFFC	GICR_CIDR3	RO	0xB1	32	Component ID 3 Register	No

Multi view access to the GICR registers

The following table shows the views that a GICR register is accessible in.

Table 5-46: Multi view access to the GICR registers

Name	View
GICR_CTLR	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_IIDR	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_TYPER	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_WAKER	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_MPAMIDR	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_PARTIDR	Not 2 or 3.
GICR_FCTLR	0
GICR_PWRR	0
GICR_CLASSR	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_VIEWR	0
GICR_FLUSHR	0
GICR_PROPBASER	Not 2 or 3.
GICR_PENDBASER	Not 2 or 3.
GICR_INVLPIR	Not 2 or 3.
GICR_INVALLR	Not 2 or 3.
GICR_SYNCR	Not 2 or 3.
GICR_MPIDR	Not 2 or 3.
GICR_PIDR4	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_PIDR5	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_PIDR6	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_PIDR7	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_PIDR0	Set by GICD_IVIEWRn or GICD_IVIEWRnE.

Name	View
GICR_PIDR1	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_PIDR2	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_PIDR3	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_CIDR0	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_CIDR1	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_CIDR2	Set by GICD_IVIEWRn or GICD_IVIEWRnE.
GICR_CIDR3	Set by GICD_IVIEWRn or GICD_IVIEWRnE.

5.4.1 GICR_CTLR, Redistributor Control Register

This register controls the operation of a Redistributor, and enables the signaling of LPIs by the Redistributor to the connected core.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-39: GICR_CTLR bit assignments

Table 5-47: GICR_CTLR bit descriptions

Bits	Name	Description	Type
[31]	UWP	Upstream write pending. Indicates whether all upstream writes have been communicated to the Distributor: **0** The effects of all upstream writes have been communicated to the Distributor. **1** Not all the effects of upstream writes have been communicated to the Distributor.	RO
[30:27]	-	Reserved, RAZ	-

Bits	Name	Description	Type
[26]	DPG1S	Disable processor selection for Group 1 Secure interrupts.	RW when GICD_TYPER.No1N == 0. RES0 when GICD_TYPER.No1N == 1.
[25]	DPG1NS	Disable processor selection for Group 1 Non-secure interrupts.	
[24]	DPG0	Disable processor selection for Group 0 interrupts.	
[23:4]	-	Reserved, RAZ	-
[3]	RWP	Register write pending: **0** No register write in progress. **1** Register write in progress.	RO
[2]	IF	Returns 1 if LPIs are supported, indicating that GICR_INVLPIR and GICR_INVALLR are implemented, else returns 0.	RO
[1]	CES	Clear enable supported. Returns 1 to indicate that software can change GICR_CTLR.EnableLPIs from 1 to 0.	RO
[0]	EnableLPIs	Controls whether LPI support is enabled: **0** LPI support is disabled. **1** LPI support is enabled. If EnableLPIs changes from 1 to 0, then the GIC flushes out all LPIs on the PE. When GICR_CTLR.RWP becomes zero, the GIC no longer accesses the Pending table of this PE. After all EnableLPIs (and RWP bits) are clear, then the GIC no longer accesses the LPI Property table.	RW

5.4.2 GICR_IIDR, Redistributor Implementation Identification Register

This register provides information about the implementer and revision of the Redistributor.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-40: GICR_IIDR bit assignments

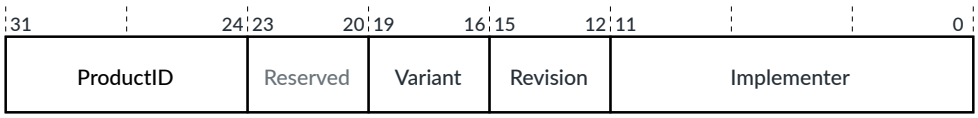


Table 5-48: GICR_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **0x07** GIC-720AE
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: **0x0** r0 **0x1** r1 **0x2** r2
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: **0x0** p0 **0x1** p1
[11:0]	Implementer	Identifies the implementer: **0x43B** Arm

5.4.3 GICR_TYPER, Redistributor Type Register

This register returns information about the features that this Redistributor supports.

Configurations

This register is available in all configurations.

Attributes

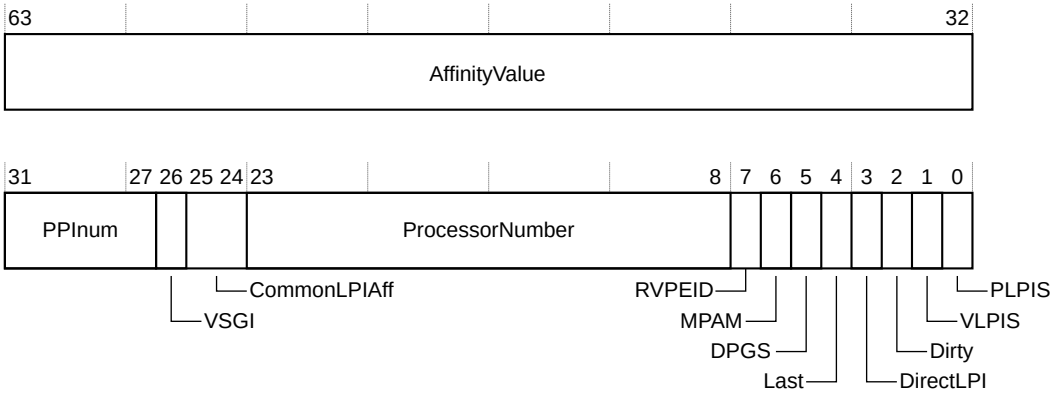
Width 64-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-41: GICR_TYPER bit assignments



In the following table, the View column is applicable only for GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1.

Table 5-49: GICR_TYPER bit descriptions

Bits	Name	Description	View
[63:32]	AffinityValue	Affinity level values for this Redistributor: **Bits[63:56], AF3** The affinity level 3 value. **Bits[55:48], AF2** The affinity level 2 value. **Bits[47:40], AF1** The affinity level 1 value. **Bits[39:32], AF0** The affinity level 0 value.	0, 1, 2, 3
[31:27]	PPInum	Indicates the maximum PPI INTID that GIC-720AE supports: **0b00000** Maximum PPI INTID is 31. **0b00001** Maximum PPI INTID is 1087.	0, 1, 2, 3
[26]	VSGI	Indicates whether this Redistributor supports direct injection of SGIs: **0** This Redistributor does not support direct injection of SGIs. This value occurs when `gicv41_support` == 0. **1** This Redistributor supports direct injection of SGIs. This value occurs when `gicv41_support` == 1.	PE view 0, 1
[25:24]	CommonLPIAff	Returns: **0b00** Single chip configuration. **0b01** If chip set by AF3. **0b10** If chip set by AF2. **0b11** Reserved. Redistributors that belong to the same CommonLPIAff group must point at the same copy of the vPE Configuration table.	0, 1, 2, 3
[23:8]	ProcessorNumber	Returns the core number and chip number that uniquely identifies this core in the system.	0, 1, 2, 3

Bits	Name	Description	View
[7]	RVPEID	Returns: **0** The GICR_VPENDBASER register does not record the index into the vPE Configuration table. This value occurs when `gicv41_support` == 0. **1** The GICR_VPENDBASER register records the index into the vPE Configuration table. This value occurs when `gicv41_support` == 1.	PE view 0, 1
[6]	MPAM	Indicates whether GIC-720AE supports *Memory Partitioning and Monitoring* (MPAM): **0** MPAM is not supported. This value occurs when `lpi_support` == 0. **1** MPAM is supported. This value occurs when `lpi_support` == 1.	PE view 0, 1
[5]	DPGS	Returns 1, to indicate that the GIC-720AE supports *Disable Processor Group Selections*. See GICR_CTLR.DPG1S, GICR_CTLR.DPG1NS, and GICR_CTLR.DPG0.	0, 1, 2, 3
[4]	Last	Last Redistributor: **0** This Redistributor is not the last Redistributor on the chip. **1** This Redistributor is the last Redistributor on the chip. When GICD_CFGID.VIEW == 1, for views 1, 2, or 3 this bit always returns 1.	0, 1, 2, 3
[3]	DirectLPI	Returns 0, to indicate that: • The GICR_INVLPIR, GICR_INVALLR, and GICR_SYNCR registers are implemented. • The GICR_SETLPIR and GICR_CLRLPIR are not implemented. The GICR_INVLPIR and GICR_INVALLR are present in all configurations of the GIC that support LPIs.	PE view 0, 1
[2]	Dirty	Returns: **0** No vLPI support. This value occurs when `gicv41_support` == 0. **1** The Redistributor sets the state of GICR_VPENDBASER.Dirty after GICR_VPROPBASER.Valid is set to 1. After every residency change, software must poll for GICR_VPENDBASER.Dirty == 0. This value occurs when `gicv41_support` == 1.	PE view 0, 1
[1]	VLPIS	Indicates whether the Redistributor supports virtual LPIs: **0** The Redistributor does not support virtual LPIs or the direct injection of virtual LPIs. This value occurs when `gicv41_support` == 0. **1** The Redistributor supports virtual LPIs and the direct injection of virtual LPIs. This value occurs when `gicv41_support` == 1. See the *Arm® Generic Interrupt Controller v3 and v4 - Virtualization*.	PE view 0, 1
[0]	PLPIS	Indicates whether the Redistributor supports physical LPIs: **0** The Redistributor does not support physical LPIs. This value occurs when `lpi_support` == 0. **1** The Redistributor supports physical LPIs. This value occurs when `lpi_support` == 1.	PE view 0, 1

5.4.4 GICR_WAKER, Power Management Control Register

This register controls whether the GIC-720AE can be powered down.

Configurations

This register is available in all configurations.

Attributes

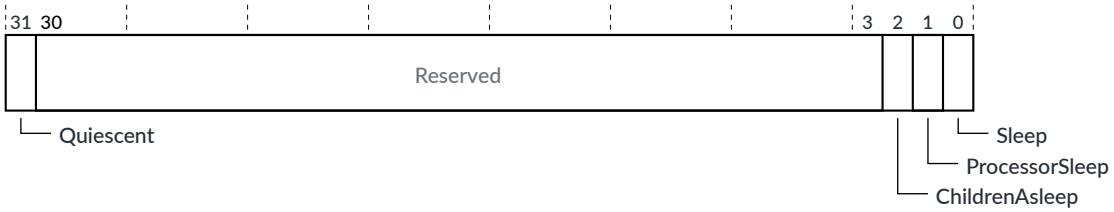
Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-42: GICR_WAKER bit assignments



In the following table, the View column is applicable only for GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1.

Table 5-50: GICR_WAKER bit descriptions

Bits	Name	Description	Type	View
[31]	Quiescent	When set to 1, it indicates that the GIC-720AE is idle and can be powered down if necessary.	RO	0, 1, 2, 3
[30:3]	-	Reserved, RAZ	-	-
[2]	ChildrenAsleep	When set to 1, it indicates that the bus between the CPU interface and this *GIC Cluster Interface* (GCI) is quiescent.	RO	0, 1, 2, 3
[1]	ProcessorSleep	Controls whether the GIC must assert a wake request signal before the GCI delivers an interrupt to the core: **0** The GIC never asserts a wake_request signal and the GCI delivers the interrupt to the core. **1** The GIC asserts a wake_request signal if there is a pending interrupt that targets the connected core. See 4.15.2 Processor core power management on page 92. If the GIC configuration supports local PE wake, then the GCI has cpu_wake_request signals. For these configurations, when a pending interrupt targets the connected core: • The GCI asserts the cpu_wake_request signal. • The Wake Request block asserts the wake_request signal. See Local PE wake on page 38.	RW	0, 1, 2, 3
[0]	Sleep	Set this bit to 1, to flush the LPI cache: **0** Normal operation. **1** The GIC-720AE ensures that all the caches are consistent with external memory and that it is safe to power down. See A.2 Other power management on page 364.	RW	0, 1

Accessibility

GICR_WAKER is accessible only by Secure accesses.

Related information

Other power management on page 364

5.4.5 GICR_MPAMIDR, Report maximum PARTID and PMG Register

This register returns the maximum values that the *Memory Partitioning and Monitoring* (MPAM) fields can be set to in GICR_PARTIDR.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-43: GICR_MPAMIDR bit assignments

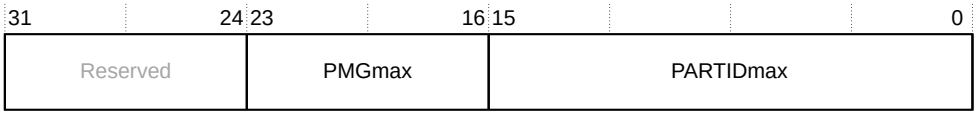


Table 5-51: GICR_MPAMIDR bit descriptions

Bits	Name	Description
[31:24]	-	Reserved
[23:16]	PMGmax	Performance monitoring group. Returns `0x01`, and indicates the maximum value that GICR_PARTIDR.PMG can be set to.
[15:0]	PARTIDmax	Returns `0x01FF`, and indicates the maximum value that GICR_PARTIDR.PARTID can be set to.

5.4.6 GICR_PARTIDR, Set PARTID and PMG Register

This register sets the Partition ID and PMG values that the Redistributor uses during memory accesses.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-44: GICR_PARTIDR bit assignments

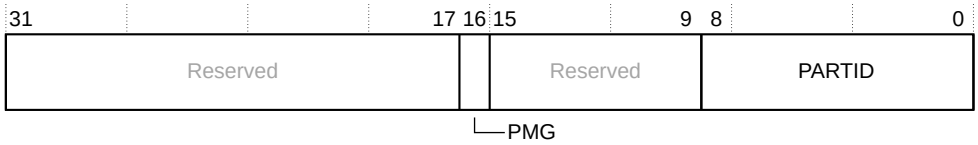


Table 5-52: GICR_PARTIDR bit descriptions

Bits	Name	Description
[31:17]	-	Reserved
[16]	PMG	The performance monitoring group value that the Redistributor uses when it accesses memory
[15:9]	-	Reserved
[8:0]	PARTID	The Partition ID value that the Redistributor uses when it accesses memory

5.4.7 GICR_FCTLR, Function Control Register

This register controls the clock gate overrides, the denial of Q-Channel requests, and the scrubbing of all RAMs in the associated Redistributor. For real-time GCIs, it can also disable the combining of GIC Stream messages.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-45: GICR_FCTLR bit assignments

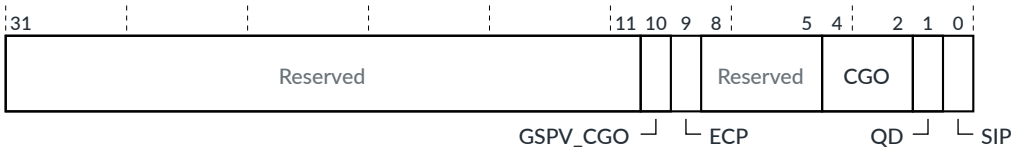


Table 5-53: GICR_FCTLR bit descriptions

Bits	Name	Description
[31:11]	-	Reserved, RAZ/WI
[10]	GSPV_CGO	Clock gate override for GSPV: **0** Use full clock gating in the GSPV. **1** Leave clock running in the GSPV. If clock gates are not implemented, then you must use this value.
[9]	ECP	Enable combined packets. This bit controls whether the Redistributor combines packets to improve the latency when it connects to Arm® Cortex®-R82 cores: **0** The Redistributor does not combine GIC Stream messages. This value occurs at reset, when `rlt` == 0. `rlt` is a configuration parameter. **1** The Redistributor combines GIC Stream messages, to improve the interrupt latency. This value occurs at reset, when `rlt` == 1. `rlt` is a configuration parameter.
[8:5]	-	Reserved, RAZ/WI
[4:2]	CGO	Clock gate override. One bit for each clock gate: **0** Use full clock gating. **1** Leave clock running. If clock gates are not implemented, then you must use this value. The clock gate bit assignments are: **Bit[4], CGO[2]** Search clock gate. **Bit[3], CGO[1]** Downstream message clock gate. **Bit[2], CGO[0]** Upstream message clock gate.
[1]	QD	Q-Channel deny: **0** Allow Q-Channel accesses. **1** Deny Q-Channel accesses.
[0]	SIP	Scrub in progress: **0** No scrub in progress. **1** Scrub in progress. This bit is read and written by software. When a scrub is complete, the GIC clears the bit to 0.

Accessibility

GICR_FCTLR is accessible only by Secure accesses.

5.4.8 GICR_PWRR, Power Register

This register controls the powerup sequence of the Redistributors. Software must write to this register during the powerup sequence.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-46: GICR_PWRR bit assignments

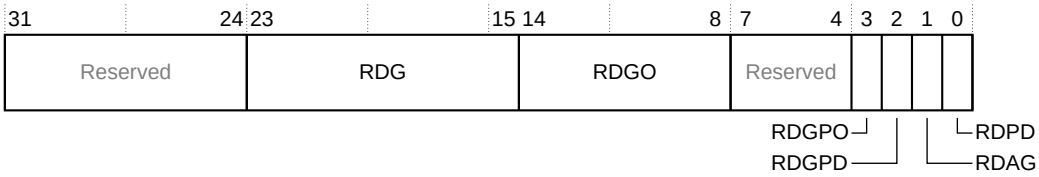


Table 5-54: GICR_PWRR bit descriptions

Bits	Name	Description	Type
[31:24]	-	Reserved, RAZ	-
[23:15]	RDG	RDGroup. This field indicates the number of the *GIC Cluster Interface* (GCI) of this Redistributor.	RO
[14:8]	RDGO	RDGroupOffset. This field indicates the identifier of the current core within the GCI.	RO
[7:4]	-	Reserved, RAZ	-
[3]	RDGPO	RDGroupPoweredOff. This bit indicates: **0** GCI is powered up and can be accessed. **1** It is safe to power down the GCI.	RO
[2]	RDGPD	RDGroupPowerDown. This bit indicates the intentional power state of the GCI: **0** Intend to power up. **1** Intend to power down. The GCI has reached its intentional power state when RDGPD = RDGPO.	RO
[1]	RDAG	RDApplyGroup. Setting this bit to 1 applies the RDPD value to all Redistributors on the same GCI. If the RDPD value cannot be applied to all cores in the group, then the GIC ignores this request.	WO

Bits	Name	Description	Type
[0]	RDPD	RDPowerDown: **0** Redistributor is powered up and can be accessed. **1** The core permits the Redistributor to be powered down. Writes to 1 are ignored if GICR_WAKER.ProcessorSleep != 1. Writes are ignored if RDGPD != RDGPO and changing to not match RDGPD. If all other cores in the Redistributor group have RDPD == 1, then setting this bit to 1 also sets RDGPD = 1.	RW

Accessibility

GICR_PWRR is accessible only by Secure accesses.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, this register is accessible only for view 0.

Related information

Redistributor power management on page 91

5.4.9 GICR_CLASSR, Class Register

This register specifies which class of 1 of N interrupt the CPU accepts.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-47: GICR_CLASSR bit assignments

Table 5-55: GICR_CLASSR bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, RAZ/WI
[0]	Class	Interrupt class: **0** Class 0 **1** Class 1

Accessibility

GICR_CLASSR is accessible only by Secure accesses.

Related information

SPI routing and 1 of N selection on page 77
GICD_ICLARn, Interrupt Class Registers on page 165

5.4.10 GICR_VIEWR, View Register

This register controls the view that this Redistributor belongs to.

Configurations

This register is available in configurations that support multi view, that is, when GICD_CFGID.VIEW == 1.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_CFGID.RDC == 0, software must write to this register to assign each PE into a view, before it writes to any other registers and before it receives messages from any PE. Otherwise the behavior is unpredictable.

If GICD_CFGID.RDC == 1, software must:

1. Write to GICD_RDOFFR<n>, if the removal of some cores is necessary.
2. Write to GICR_MPIDR, if changes to the affinity values are necessary.
3. Write to GICR_VIEWR to assign PEs to a view. Software must complete the writes to GICR_VIEWR registers, before it writes to any other registers and before it receives messages from any PE. Otherwise the behavior is unpredictable.

Bit descriptions

Figure 5-48: GICR_VIEWR bit assignments

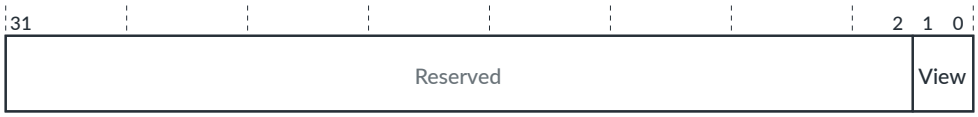


Table 5-56: GICR_VIEWR bit descriptions

Bits	Name	Description
[31:2]	-	Reserved, RAZ/WI
[1:0]	View	Controls which view this Redistributor is assigned to: **0b00** This Redistributor is assigned to view 0. **0b01** This Redistributor is assigned to view 1. **0b10** This Redistributor is assigned to view 2. **0b11** This Redistributor is assigned to view 3.

Accessibility

GICR_VIEWR is accessible only by Secure accesses from view 0.

5.4.11 GICR_FLUSHR, Flush Register

This register controls the recovery mode for the *GIC Stream Protocol Validator* (GSPV) in the GCI.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

Software must only change bits[29:26] when all PEs are asleep for this GCI, otherwise unpredictable behavior might occur.

Bit descriptions

Figure 5-49: GICR_FLUSHR bit assignments

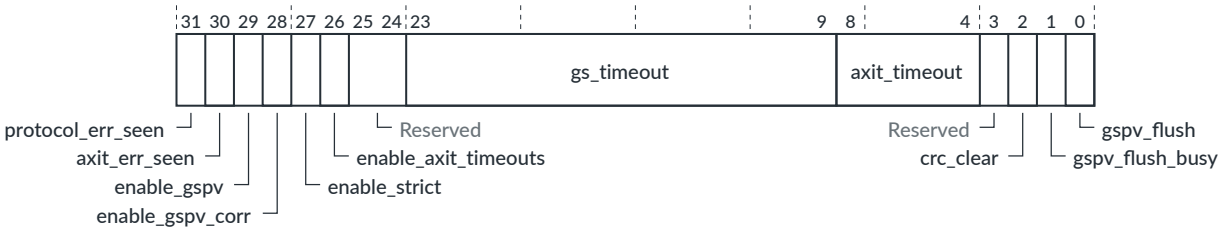


Table 5-57: GICR_FLUSHR bit descriptions

Bits	Name	Description	Type
[31]	protocol_err_seen	For each core, indicates whether the GSPV has detected a GIC Stream protocol violation: **0** The GSPV has not detected a GIC Stream protocol violation. **1** The GSPV has detected a GIC Stream protocol violation.	R/W1C
[30]	axit_err_seen	For each core, indicates whether the GSPV has detected an AXI5-Stream protocol violation: **0** The GSPV has not detected an AXI5-Stream protocol violation. **1** The GSPV has detected an AXI5-Stream protocol violation.	R/W1C
[29]	enable_gspv	For this GCI, this bit enables or disables the GSPV: **0** The GSPV is not enabled. **1** The GSPV is enabled. This value occurs at reset.	R/W
[28]	enable_gspv_corr	For this GCI, this bit enables or disables the protocol correction in the GSPV: **0** The GSPV protocol correction is not enabled. **1** The GSPV protocol correction is enabled. This value occurs at reset. To protect against lower safety-level cores in a mixed criticality use case, software must set both the enable_gspv and enable_gspv_corr bits to 1.	R/W
[27]	enable_strict	Controls whether the GSPV detects protocol violations that indicate hardware errors that would not affect other PEs on the GIC, but do indicate a hardware issue with the issuing PE: **0** Strict checks are not enabled. **1** Strict checks are enabled. This value occurs at reset.	R/W
[26]	enable_axit_timeouts	Controls whether timeouts on the iritready signal and the interbeat timings are tracked: **0** The GSPV does not monitor any timeouts of the iritready signal and the interbeat timings. **1** The GSPV monitors for any timeouts of the iritready signal and the interbeat timings. This value occurs at reset.	R/W
[25:24]	-	Reserved	-
[23:9]	gs_timeout	Sets the timeout value for the GIC Stream and the iritready signal checks on this GCI. The timeout is 8192 × (gs_timeout + 1) − 1 cycles. The value at reset is `0xEFF`.	R/W

Bits	Name	Description	Type
[8:4]	axit_timeout	Sets the timeout value for the AXI5-Stream interbeat timing checks on this GCI. The timeout is 256 × (axit_timeout + 1) − 1 cycles. The value at reset is `0x1F`.	R/W
[3]	-	Reserved	-
[2]	crc_clear	Set to 1 to reset the GIC Stream protection CRC state on this GCI.	WO
[1]	gspv_flush_busy	For this GCI, indicates whether a flush is occurring on any core: **0** No flush is occuring on a core that connects to this GCI. **1** A flush is occuring on a core that connects to this GCI.	RO
[0]	gspv_flush	For this core, starts a flush or returns the flush status: **0** No flush is occuring on this core. **1** A flush is occuring on this core.	RW

Accessibility

GICR_FLUSHR is accessible only by Secure accesses.

If GICD_CFGID.VIEW == 1, then GICR_FLUSHR is accessible only for view 0.

5.4.12 GICR_MPIDR, MPIDR Register

This register allows Secure software to write the affinity values of a Redistributor.

Configurations

This register is available in configurations when GICD_CFGID.RDC == 1.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

Software must program this register after it writes to the GICD_RDOFFRn registers and before the GIC receives messages from any processors or any other register accesses. Otherwise the behavior is unpredictable.

Programming of GICR_MPIDR must be unique for each Redistributor. If multi view is supported, that is GICD_CFGID.VIEW == 1, then GICR_MPIDR needs to be unique only within the view.

Bit descriptions

Figure 5-50: GICR_MPIDR bit assignments

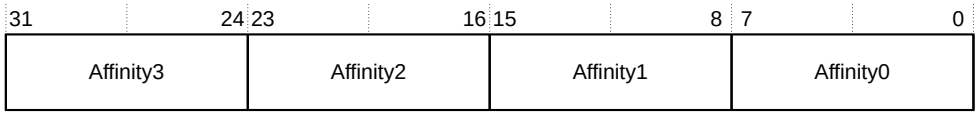


Table 5-58: GICR_MPIDR bit descriptions

Bits	Name	Description
[31:24]	Affinity3	Sets the affinity level 3 value of this Redistributor. The `max_affinity_width3` configuration parameter controls how many of the lower bits are implemented. This field ignores writes for cross-chip configurations or when `max_affinity_width3` is zero. Software can use GICR_TYPER.AffinityValue to read the affinity level 3 value.
[23:16]	Affinity2	Sets the affinity level 2 value of this Redistributor. The `max_affinity_width2` configuration parameter controls how many of the lower bits are implemented. This field ignores writes for cross-chip configurations with chip affinity level 2 or when `max_affinity_width2` is zero. Software can use GICR_TYPER.AffinityValue to read the affinity level 2 value.
[15:8]	Affinity1	Sets the affinity level 1 value of this Redistributor. The `max_affinity_width1` configuration parameter controls how many of the lower bits are implemented. This field ignores writes when `max_affinity_width1` is zero. Software can use GICR_TYPER.AffinityValue to read the affinity level 1 value.
[7:0]	Affinity0	Sets the affinity level 0 value of this Redistributor. The `max_affinity_width0` configuration parameter controls how many of the lower bits are implemented. This field ignores writes when `max_affinity_width0` is zero. Software can use GICR_TYPER.AffinityValue to read the affinity level 0 value.

Accessibility

GICR_MPIDR is accessible only by Secure accesses.

If GICD_CFGID.VIEW == 1, then GICR_MPIDR is accessible only for view 0.

5.4.13 GICR_PIDR2, Peripheral ID2 Register

This register returns byte[2] of the peripheral ID. The GICR_PIDR2 register is part of the set of Redistributor peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.4 Redistributor registers for control and physical LPIs summary on page 188 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-51: GICR_PIDR2 bit assignments

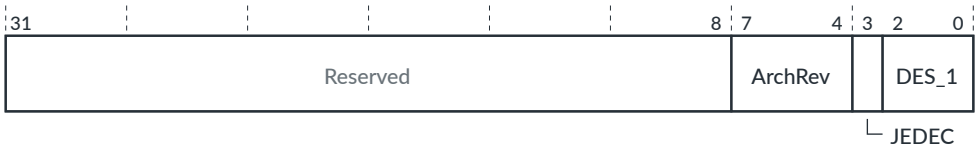


Table 5-59: GICR_PIDR2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:4]	ArchRev	Identifies the version of the GIC architecture with which the Redistributor complies: **0x3** GICv3 **0x4** GICv4
[3]	JEDEC	Indicates that a JEDEC-assigned JEP106 identity code is used.
[2:0]	DES_1	Bits[6:4] of the JEP106 identity code. Bits[3:0] of the JEP106 identity code are assigned to GICR_PIDR1[7:4].

5.5 Redistributor registers for SGIs and PPIs summary

The functions for the GIC-720AE SGIs and PPIs are controlled through the Redistributor registers identified with the prefix GICR.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, these GICR registers are accessible for view 0 and the view that GICR_VIEWR sets.

For descriptions of registers that are not specific to the GIC-720AE, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Table 5-60: Redistributor registers for SGIs and PPIs summary

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0000-0x007C	-	-	-	-	Reserved	-
0x0080	GICR_IGROUPR0	RW	0x0	32	Interrupt Group Register	Yes
0x0084	GICR_IGROUPR1E	RW	0x0	32	Interrupt Group Register Extended. Only present when ppis_per_cpu > 16.	Yes
0x0088-0x00FC	-	-	-	-	Reserved	-
0x0100	GICR_ISENABLER0	RW	0x0	32	Interrupt Set-Enable Register	Yes
0x0104	GICR_ISENABLER1E	RW	0x0	32	Interrupt Set-Enable Register Extended. Only present when ppis_per_cpu > 16.	Yes
0x0108-0x017C	-	-	-	-	Reserved	-

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0180	GICR_ICENABLER0	RW	0x0	32	Interrupt Clear-Enable Register	Yes
0x0184	GICR_ICENABLER1E	RW	0x0	32	Interrupt Clear-Enable Register Extended. Only present when ppis_per_cpu > 16.	Yes
0x0188-0x01FC	-	-	-	-	Reserved	-
0x0200	GICR_ISPENDR0	RW	PPI signal dependent	32	Interrupt Set-Pending Register	Yes
0x0204	GICR_ISPENDR1E	RW	PPI signal dependent	32	Interrupt Set-Pending Register Extended. Only present when ppis_per_cpu > 16.	Yes
0x0208-0x027C	-	-	-	-	Reserved	-
0x0280	GICR_ICPENDR0	RW	PPI signal dependent	32	Peripheral Clear Pending Register	Yes
0x0284	GICR_ICPENDR1E	RW	PPI signal dependent	32	Peripheral Clear-Pending Register Extended. Only present when ppis_per_cpu > 16.	Yes
0x0288-0x02FC	-	-	-	-	Reserved	-
0x0300	GICR_ISACTIVER0	RW	0x0	32	Interrupt Set-Active Register	Yes
0x0304	GICR_ISACTIVER1E	RW	0x0	32	Interrupt Set-Active Register Extended. Only present when ppis_per_cpu > 16.	Yes
0x0308-0x037C	-	-	-	-	Reserved	-
0x0380	GICR_ICACTIVER0	RW	0x0	32	Interrupt Clear-Active Register	Yes
0x0384	GICR_ICACTIVER1E	RW	0x0	32	Interrupt Clear-Active Register Extended. Only present when ppis_per_cpu > 16.	Yes
0x0388-0x03FC	-	-	-	-	Reserved	-
0x0400-0x041C	GICR_IPRIORITYRn	RW	0x0	32	Interrupt Priority Registers	Yes
0x0420	GICR_IPRIORITYRnE	RW	0x0	32	Interrupt Priority Registers Extended. Only present when ppis_per_cpu > 16.	Yes
0x0440-0x0BFC	-	-	-	-	Reserved	-
0x0C00-0x0C04	GICR_ICFGRn	RW	0xAAAAAAAA when n == 0. 0x0 when n == 1.	32	Interrupt Configuration Registers	Yes
0x0C08-0x0C0C	GICR_ICFGRnE	RW	0x0	32	Interrupt Configuration Registers Extended. Only present when ppis_per_cpu > 16.	Yes
0x0C10-0x0CFC	-	-	-	-	Reserved	-
0x0D00	GICR_IGRPMODR0	RW	0x0	32	Interrupt Group Modifier Register	Yes
0x0D04-0x0C0C	GICR_IGRPMODR1E	RW	0x0	32	Interrupt Group Modifier Register Extended. Only present when ppis_per_cpu > 16.	Yes
0x0D08-0x0DFC	-	-	-	-	Reserved	-

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0E00	GICR_NSACR	RW	0x0	32	Non-secure Access Control Register	Yes
0x0E04-0xBFFC	-	-	-	-	Reserved	-
0xC000	GICR_MISCSTATUSR	RO	0x0	32	Miscellaneous Status Register	No
0xC004	-	-	-	-	Reserved	-
0xC008	GICR_ICDERRR	RW	0x0	32	Interrupt Clear Distribution Error Register	No
0xC00C	-	-	-	-	Reserved	-
0xC010	GICR_SGIDR	RW	-	64	SGI Default Register	No
0xC018	GICR_DPRIR	RW	0x0	32	Default Priority Register	No
0xC01C-0xC0FC	-	-	-	-	Reserved	-
0xC100	GICR_ICERRR0	RW	0x0	32	Interrupt Clear Error Register	
0xC104	GICR_ICERRR1E	RW	0x0	32	Interrupt Clear Error Register Extended. Only present when ppis_per_cpu > 16.	
0xC108-0xC17C	-	-	-	-	Reserved	-
0xC180	GICR_ISERRR0	RW	0x0	32	Interrupt Set Error Register	No
0xC184	GICR_ISERRR1E	RW	0x0	32	Interrupt Set Error Register Extended. Only present when ppis_per_cpu > 16.	No
0xC188-0xEFFC	-	-	-	-	Reserved	-
0xF000	GICR_CFGID0	RO	Configuration dependent	32	Configuration ID0 Register	No
0xF004	GICR_CFGID1	RO	Configuration dependent	32	Configuration ID1 Register	No
0xF010	GICR_ERRINSR	RW	0x0	64	Error Insertion Register	No

5.5.1 GICR_MISCSTATUSR, Miscellaneous Status Register

Use this register to test the integration of the cpu_active and wake_request input signals. You can also use the register to debug the CPU interface enables that GIC-720AE observes.

Bits[2:0] are a copy of the CPU interface group enables for the core corresponding to this Redistributor. These copies are undefined when ProcessorSleep or ChildrenAsleep is set for a core, because the core is presumed to be powered down. Upstream write packets maintain these copies that can de-synchronize after an incorrect powerdown sequence. This register enables you to debug this scenario. For more information, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-52: GICR_MISCSTATUSR bit assignments

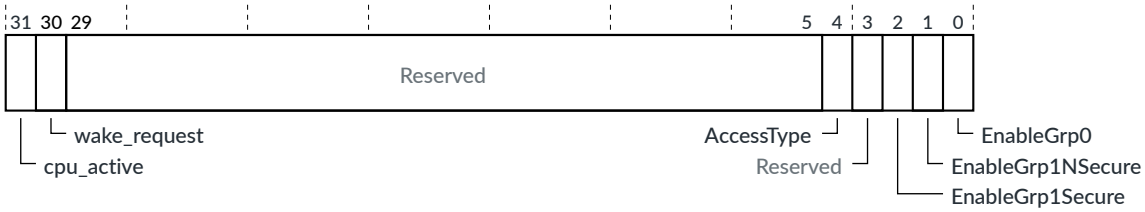


Table 5-61: GICR_MISCSTATUSR bit descriptions

Bits	Name	Description
[31]	cpu_active	Returns the status of the cpu_active signal for the core corresponding to the Redistributor whose register is being read: **0** cpu_active input signal is not active. **1** cpu_active input signal is active. This bit is undefined when ProcessorSleep or ChildrenAsleep is set for a core, because the core is presumed to be powered down.
[30]	wake_request	Returns the status of the wake_request signal: **0** wake_request signal is not active. **1** wake_request signal is asserted.
[29:5]	-	Reserved
[4]	AccessType	Returns the access type: **0** Secure access. If GICD_CTLR.DS == 1, then this bit returns 0. **1** Non-secure access
[3]	-	Reserved
[2]	EnableGrp1Secure	In systems that enable two Security states, when GICD_CTLR.DS == 0, then: • For Secure reads, returns the Group 1 Secure CPU interface enable. • For Non-secure reads, returns zero. In systems that only enable a single Security state, when GICD_CTLR.DS == 1, then this bit returns zero.

Bits	Name	Description
[1]	EnableGrp1NSecure	In systems that enable two Security states, when GICD_CTLR.DS == 0, then: • For Secure reads, this bit returns the Group 1 Non-secure CPU interface enable. • For Non-secure reads, when GICD_CTLR.ARE_NS == 1, this bit returns the Group 1 Non-secure CPU interface enable. • For Non-secure reads when GICD_CTLR.ARE_NS == 0, this bit returns zero. In systems that only enable a single Security state, when GICD_CTLR.DS == 1, this bit returns the Group 1 CPU interface enable.
[0]	EnableGrp0	In systems that enable two Security states, when GICD_CTLR.DS == 0, then: • For Secure reads, this bit returns the Group 0 CPU interface enable. • For Non-secure reads when GICD_CTLR.ARE_NS == 0, this bit returns the Group 1 Non-secure CPU interface enable. • For Non-secure reads when GICD_CTLR.ARE_NS == 1, this bit returns zero. In systems that only enable a single Security state, when GICD_CTLR.DS == 1, this bit returns the Group 0 CPU interface enable.

5.5.2 GICR_ICDERRR, Interrupt Clear Distribution Error Register

This register indicates if the SGI distribution data has been corrupted in SRAM. You can use this register to clear an SGI error.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-53: GICR_ICDERRR bit assignments

Table 5-62: GICR_ICDERRR bit descriptions

Bits	Name	Description
[31:16]	-	Reserved

Bits	Name	Description
[15:0]	Error	Indicates whether an SGI is in an error state: **Bit[n] = 0** If read, SGIn is not in an error state. Writing 0 has no effect. **Bit[n] = 1** If read, SGIn is in an error state, so the interrupt is not delivered. Writing 1 clears the error on SGIn.

Accessibility

GICR_ICDERRR is accessible only by Secure accesses.

5.5.3 GICR_SGIDR, SGI Default Register

This register controls the default value of SGI settings, for use in the case of a *Double-bit Error Detect Error* (DEDERR).

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, this register is banked for each view.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-63: GICR_SGIDR bit descriptions

Bits	Name	Description
[3] + 4n: [63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3]	-	Reserved, RES0
[2] + 4n: [62, 58, 54, 50, 46, 42, 38, 34, 30, 26, 22, 18, 14, 10, 6, 2]	GRPMOD	As GICR_IGRPMODR0 register.
[1] + 4n: [61, 57, 53, 49, 45, 41, 37, 33, 29, 25, 21, 17, 13, 9, 5, 1]	GRP	As GICR_IGROUPR0 register.
[0] + 4n: [60, 56, 52, 48, 44, 40, 36, 32, 28, 24, 20, 16, 12, 8, 4, 0]	NSACR	1 = Allow Non-secure access to interrupt <n>.

Accessibility

GICR_SGIDR is accessible only by Secure accesses.

5.5.4 GICR_DPRIR, Default Priority Register

This register controls the default priority of errored interrupts.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, this register is banked for each view.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-54: GICR_DPRIR bit assignments

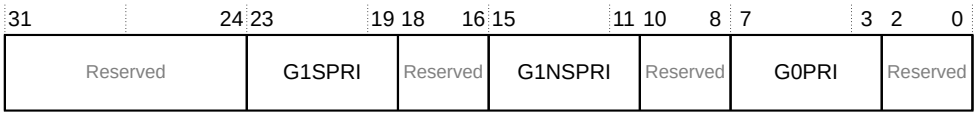


Table 5-64: GICR_DPRIR bit descriptions

Bits	Name	Description
[31:24]	-	Reserved, **RES0**
[23:19]	G1SPRI	The default priority that the GIC uses for errored Secure Group 1 interrupts. Lower priority values correspond to greater priority of the interrupt. Only Secure writes can update this field.
[18:16]	-	Reserved, **RES0**
[15:11]	G1NSPRI	The default priority that the GIC uses for errored Non-secure Group 1 interrupts. Lower priority values correspond to greater priority of the interrupt.
[10:8]	-	Reserved, **RES0**
[7:3]	G0PRI	The default priority that the GIC uses for errored Group 0 interrupts. Lower priority values correspond to greater priority of the interrupt. Only Secure writes can update this field.
[2:0]	-	Reserved, **RES0**

Accessibility

Some fields are writable only by using a Secure access.

5.5.5 GICR_ICERRR0, Interrupt Clear Error Register 0

This register indicates if the SGI or PPI data has been corrupted in the GCI RAM. Software can use this register to clear an SGI or PPI error.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-55: GICR_ICERRR0 bit assignments

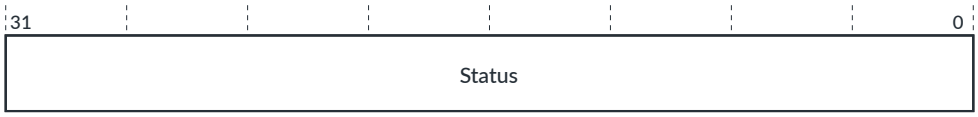


Table 5-65: GICR_ICERRR0 bit descriptions

Bits	Name	Description
[31:16]	Status	Indicates whether a PPI is in an error state: **Bit[n] = 0** If read, PPI[n–16] is not in an error state. Writing 0 has no effect. **Bit[n] = 1** If read, PPI[n–16] is in an error state, so the interrupt is not delivered. Writing 1 clears the error on PPI[n–16].
[15:0]		Indicates whether an SGI is in an error state: **Bit[n] = 0** If read, SGI[n] is not in an error state. Writing 0 has no effect. **Bit[n] = 1** If read, SGI[n] is in an error state, so the interrupt is not delivered. Writing 1 clears the error on SGI[n].

Accessibility

GICR_ICERRR0 is accessible only by Secure accesses.

5.5.6 GICR_ICERRR1E, Interrupt Clear Error Register Extended

This register indicates if the PPI[47:16] data has been corrupted in the GCI RAM. Software can use this register to clear an error.

Configurations

This register available in configurations with > 16 PPIs, that is, when GICR_TYPER.PPInum >0.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-56: GICR_ICERRR1E bit assignments

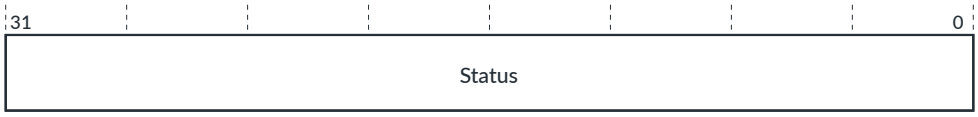


Table 5-66: GICR_ICERRR1E bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether a PPI[47:16] is in an error state: **Bit[n] = 0** If read, PPI[n+16] is not in an error state. Writing 0 has no effect. **Bit[n] = 1** If read, PPI[n+16] is in an error state, so the interrupt is not delivered. Writing 1 clears the error on PPI[n+16].

Accessibility

GICR_ICERRR1E is accessible only by Secure accesses.

5.5.7 GICR_ISERRR0, Interrupt Set Error Register 0

This register indicates if the SGI or PPI data has been corrupted in the GCI RAM. For testing purposes, software can use this register to set an SGI or PPI error.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-57: GICR_ISERRR0 bit assignments

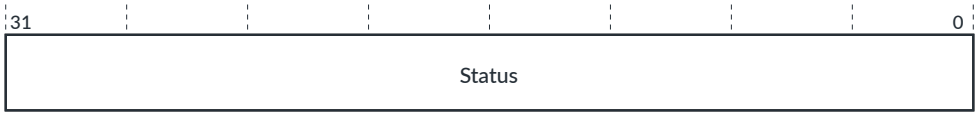


Table 5-67: GICR_ISERRR0 bit descriptions

Bits	Name	Description
[31:16]	Status	Indicates whether a PPI is in an error state: **Bit[n] = 0** If read, PPI[n–16] is not in an error state. Writing 0 has no effect. **Bit[n] = 1** If read, PPI[n–16] is in an error state, so the interrupt is not delivered. Writing 1 sets the error on PPI[n–16].
[15:0]		Indicates whether an SGI is in an error state: **Bit[n] = 0** If read, SGI[n] is not in an error state. Writing 0 has no effect. **Bit[n] = 1** If read, SGI[n] is in an error state, so the interrupt is not delivered. Writing 1 sets the error on SGI[n].

Accessibility

GICR_ISERRR0 is accessible only by Secure accesses.

5.5.8 GICR_ISERRR1E, Interrupt Set Error Register Extended

This register indicates if the PPI[47:16] data has been corrupted in the GCI RAM. For testing purposes, software can use this register to set a PPI error.

Configurations

This register is available in configurations with > 16 PPIs, that is, when GICR_TYPER.PPInum >0.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-58: GICR_ISERRR1E bit assignments

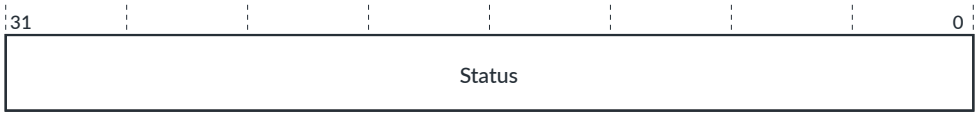


Table 5-68: GICR_ISERRR1E bit descriptions

Bits	Name	Description
[31:0]	Status	Indicates whether a PPI[47:16] is in an error state: **Bit[n] = 0** If read, PPI[n+16] is not in an error state. Writing 0 has no effect. **Bit[n] = 1** If read, PPI[n+16] is in an error state, so the interrupt is not delivered. Writing 1 sets the error on PPI[n+16].

Accessibility

GICR_ISERRR1E is accessible only by Secure accesses.

5.5.9 GICR_CFGID0, Configuration ID0 Register

This register returns information about the configuration of the Redistributors.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-59: GICR_CFGID0 bit assignments

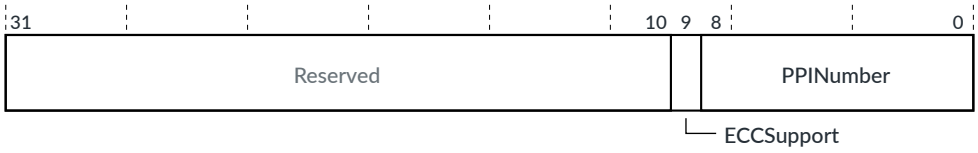


Table 5-69: GICR_CFGID0 bit descriptions

Bits	Name	Description
[31:10]	-	Reserved, RAZ
[9]	ECCSupport	1 = ECC is supported.
[8:0]	PPINumber	RedistributorID. The ppi_id[15:0] tie-off signal sets the value of the ID. Each Redistributor must have a unique ID.

5.5.10 GICR_CFGID1, Configuration ID1 Register

This register returns information about the configuration of the Redistributors.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-60: GICR_CFGID1 bit assignments

Table 5-70: GICR_CFGID1 bit descriptions

Bits	Name	Description
[31:28]	Version	Identifies the major and minor revisions of GIC-720AE: **0x0** r0p0 **0x1** r0p1 **0x2** r1p0 **0x3** r2p0
[27:24]	UserValue	Modification value that you can set. Indicates whether the customer has modified the behavior of the Redistributor. Usually, this field is `0x0`. Customers change this value when they make authorized modifications to the Redistributor.
[23:16]	PPIs_per_Processor	The number of PPIs for each core. The possible values are: • `0b0001_0000`, 16 PPIs • `0b0010_0000`, 32 PPIs • `0b0011_0000`, 48 PPIs
[15:12]	-	Reserved
[11:4]	NumCPUs	The number of cores that this Redistributor supports. GIC-720AE supports up to 64 cores, so the maximum value of this field is `0x3F`.
[3:0]	-	Reserved, RAZ

5.5.11 GICR_ERRINSR, Error Insertion Register

This register can inject errors into the PPI RAM. You can use this register to test your error recovery software.

Configurations

This register is available in configurations where the *GIC Cluster Interface* (GCI) supports ECC. See Limitations on page 363 for information about situations where the GICRn_ERRINSR register is not present.

Attributes

Width 64-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 207 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

The bit assignments within this register depend on whether a write access or read access occurs.

The following table shows the bit assignments for write accesses.

Table 5-71: GICR_ERRINSR bit assignments for writes

Bits	Name	Description
[63]	Valid	Set to 1, to start the error injection process. The GIC sets this bit to 0 when it completes the process.
[62:61]	-	**RES0**
[60]	DisableWriteCheck	Controls whether to include an encoding check: **0** Include an encoder check. **1** Disable an encoder check.
[59:48]	-	**RES0**
[47:32]	ADDR	Address
[31]	ERRINS2VALID	Controls whether the second error is valid: **0** The ERRINS2LOC field is not valid. **1** The ERRINS2LOC field is valid.
[30:25]	-	**RES0**
[24:16]	ERRINS2LOC	Sets the address location of the second error
[15]	ERRINS1VALID	Controls whether the first error is valid: **0** The ERRINS1LOC field is not valid. **1** The ERRINS1LOC field is valid.
[14:9]	-	**RES0**
[8:0]	ERRINS1LOC	Sets the address location of the first error.

The following table shows the bit assignments for read accesses.

Table 5-72: GICR_ERRINSR bit assignments for reads

Bits	Name	Description
[63]	Valid	Indicates if the error injection process is complete: **0** Error injection process is complete. **1** Error injection process is in progress.
[62:61]	Status	Indicates if the error injection process was successful, and the value is valid only when Valid == 0: **0b00** The GIC performed the error injection process. **0b01** An out-of-range error occurred. To fix this error, check that the RAM ID and the error locations are correct. **0b10** A coincident error occurred. **0b11** An encoder or decoder mismatch occurred.
[60]	RAM_Present	Indicates whether a RAM with ECC is present: **1** RAM with ECC is present.
[59:48]	-	**RES0**
[47:32]	RAM_MAX	Returns the maximum address of the RAM.
[31:9]	-	**RES0**
[8:0]	RAM WIDTH	Returns the highest maximum bit width of the RAM. For example, a value of 15 indicates a 16-bit wide RAM.

Accessibility

If GICD_SAC.GICTNS == 0, then GICR_ERRINSR is accessible only by Secure accesses.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, GICR_ERRINSR is accessible only from view 0.

5.6 vLPI register summary

The functions for the GIC-720AE vLPIs are controlled through the Redistributor registers identified with the prefix GICR.

This page does not exist in GIC-720AE configurations that do not support vLPIs.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, these GICR registers are accessible for view 0 and the view that GICR_VIEWR sets.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for information about the vLPI registers.

Table 5-73: vLPI register summary

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0000-0x006C	-	-	-	-	Reserved	-
0x0070	GICR_VPROPBASER	RW	-	64	Virtual Redistributor Properties Base Address Register	Yes
0x0078	GICR_VPENDBASER	RW	-	64	Virtual Pending Table Base Address Register	Yes
0x007C	-	-	-	-	Reserved	-
0x0080	GICR_VSGIR	WO	-	32	Virtual SGI Register	Yes
0x0084	-	-	-	-	Reserved	-
0x0088	GICR_VSGIPENDR	RO	0x0	32	Virtual SGI Pending Register	Yes
0x008C-0xBFFC	-	-	-	-	Reserved	-
0xC000	GICR_VFCTLR	RW	0x0	32	Virtual Function Control Register	No
0xC004-0xC0FC	-	-	-	-	Reserved	-
0xC100	GICR_VCFGBASER	RO	0x0	64	vICM Final vPE CFG Attribute Register	No
0xC108-0xC11C	-	-	-	-	Reserved	-
0xC120	GICR_VINVCHIPR	RW	0	32	vPE Invalidate Chip Register	No
0xC124-0xE0FC	-	-	-	-	Reserved	-
0xE100	GICR_VERRR	RW	0x0	64	vICM vPE Error Register	No
0xE108-0xFFFC	-	-	-	-	Reserved	-

5.6.1 GICR_VFCTLR, Virtual Function Control Register

This register controls the chicken bit functionality in the vICM. You can use GICR_VFCTLR to restrict the vLPI and vSGI buffer size to 1, and restrict the number of cross-chip vSGI tokens.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.6 vLPI register summary on page 220 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-61: GICR_VFCTLR bit assignments

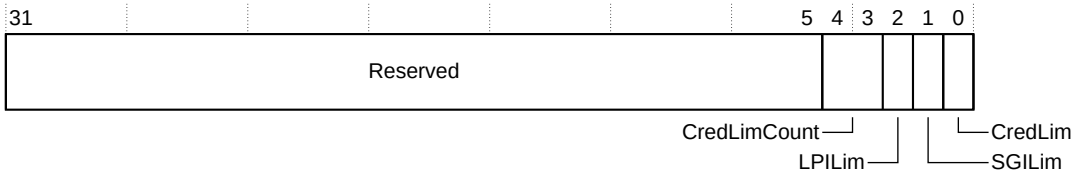


Table 5-74: GICR_VFCTLR bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, **RES0**
[4:3]	CredLimCount	When CredLim == 1, this field can reduce the number of vSGIs that can be sent to each chip: **0** 1 vSGI can be outstanding to each chip. **1** 2 vSGIs can be outstanding to each chip. **2** 3 vSGIs can be outstanding to each chip. **3** 4 vSGIs can be outstanding to each chip. If you set a value that is greater than `vsgi_cc_tokens` – 1, then the GIC behaves as if CredLim == 0.
[2]	LPILim	When set to 1, limits vLPI buffer size to 1.
[1]	SGILim	When set to 1, limits vSGI buffer size to 1.
[0]	CredLim	This bit enables you to reduce the number of vSGIs that can be sent to each chip: **0** The `vsgi_cc_tokens` configuration parameter sets the number of vSGIs that can be sent to each chip. **1** The CredLimCount field sets the number of vSGIs that can be sent to each chip.

Accessibility

GICR_VFCTLR is accessible only by Secure accesses.

5.6.2 GICR_VCFGBASER, vICM Final vPE CFG Attribute Register

This register returns the access attributes of the vPE Configuration table.

Configurations

This register is available in all configurations that support vLPIs.

Attributes

Width 64-bit
Functional group See 5.6 vLPI register summary on page 220 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-62: GICR_VCFGBASER bit assignments

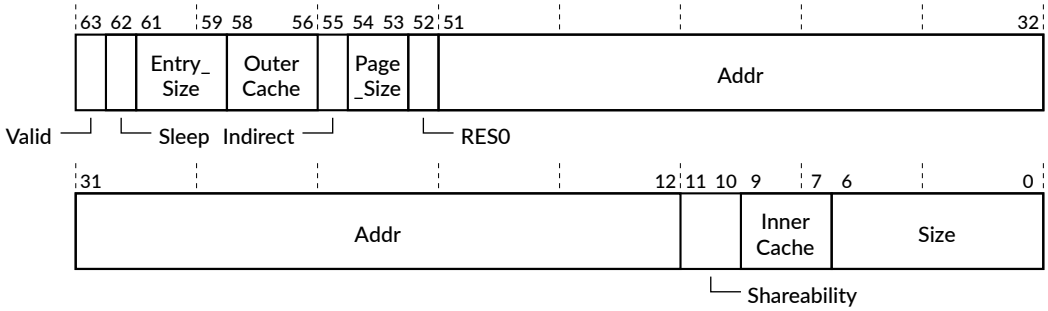


Table 5-75: GICR_VCFGBASER bit descriptions

Bits	Name	Description
[63]	Valid	Indicates whether the access attributes of the vPE Configuration table are valid: **0** The access attributes of the vPE Configuration table are not valid. **1** The access attributes of the vPE Configuration table are valid.
[62]	Sleep	Returns the value of GICR_WAKER.Sleep
[61:59]	Entry_Size	Returns the value of GICR_VPROPBASER.Entry_Size
[58:56]	OuterCache	Returns the value of GICR_VPROPBASER.OuterCache
[55]	Indirect	Returns the value of GICR_VPROPBASER.Indirect
[54:53]	Page_Size	Returns the value of GICR_VPROPBASER.Page_Size
[52]	-	**RES0**
[51:12]	Addr	Returns bits[51:12] of the vPE Configuration table base address
[11:10]	Shareability	Returns the value of GICR_VPROPBASER.Shareability
[9:7]	InnerCache	Returns the value of GICR_VPROPBASER.InnerCache
[6:0]	Size	Returns the value of GICR_VPROPBASER.Size

5.6.3 GICR_VINVCHIPR, vPE Invalidate Chip Register

This register can invalidate the vICM RAM in selected chips.

Configurations

This register is available in all configurations that support vLPIs.

Attributes

Width 32-bit

Functional group See 5.6 vLPI register summary on page 220 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-63: GICR_VINVCHIPR bit assignments

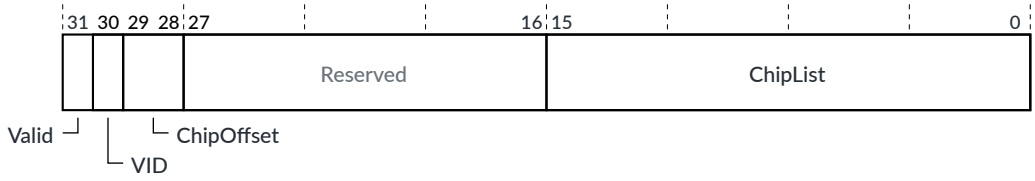


Table 5-76: GICR_VINVCHIPR bit descriptions

Bits	Name	Description
[31]	Valid	Set to 1, to start an invalidation request to the vPEs in the chips that GICR_VINVCHIPR.ChipList selects. When read as 0, it indicates that the invalidate request is complete.
[30]	VID	Virtual invalidate disable. If set to 1, the invalidation request does not invalidate the vICM RAM, but the GIC performs a drain of vLPIs, vSGIs, commands and streams. RAZ/WI in single chip configurations.
[29:28]	ChipOffset	When written, this field controls which range of chips that GICR_VINVCHIPR.ChipList selects: **0b00** Chips 0-15 **0b01** Chips 16-31 **0b10** Chips 32-47 **0b11** Chips 48-63 For reads, returns **RES0**.
[27:16]	-	**RES0**
[15:0]	ChipList	When one or more bits are set to 1, it selects a list of chips. Only vPEs with chip information set in this field are invalidated in RAM. The GIC ignores writes that attempts to select a chip that exceeds the number of configured chips. For reads, returns **RES0**.

5.6.4 GICR_VERRR, vICM vPE Error Register

This register can set and clear the error bit for a vPE in the vICM RAM. You can use the register to find vPEs with an error in the vICM and obtain vPE information from the vTGT cache and the vICM.

Configurations

This register is available in all configurations that support vLPIs.

Attributes

Width 64-bit
Functional group See 5.6 vLPI register summary on page 220 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

The bit assignments within this register can change, depending on whether you are initiating a request or reading the information of a read (RD) request.

The following table shows the bit assignments when initiating a request.

Table 5-77: GICR_VERRR bit assignments, for request initiation

Bits	Name	Description	Type
[63]	Busy	Set to 1, to start a request. The GIC sets this bit to 0 when it completes the request.	RW
[62]	Response	This bit indicates if the request was successful, and is valid only when Busy == 0: **0** The GIC performed the request. **1** The GIC failed to perform the request.	RO
[61:60]	Opcode	Request type: **0** RD. Read vPE information. **1** SET. Set the error bit. **2** CLR. Clear the error bit. **3** FIND. Find a vPE that contains an error.	RW
[59:17]	-	**RES0**	-
[16:14]	Read_block	Controls which data to retrieve for an RD operation (Opcode == 0): **0** Doorbell data. See Table 5-78: GICR_VERRR bit assignments, for a Doorbell read request on page 226. **1** vPT data. See Table 5-79: GICR_VERRR bit assignments, for a vPT read request on page 226. **2, 5-7** vCONF data. See Table 5-80: GICR_VERRR bit assignments, for a vCONF read request on page 227. **3** vSGI[15:8] programming. See Table 5-81: GICR_VERRR bit assignments, for a vSGI read request on page 228. **4** vSGI[7:0] programming. See Table 5-81: GICR_VERRR bit assignments, for a vSGI read request on page 228.	RW
[13:n]	-	**RES0**	-
[n–1:0]	vPEID	For RD, SET, and CLR requests (Opcode ≤ 2), this field selects the vPE that receives the request. For FIND requests (Opcode == 3), this field selects the vPE where the error search starts. If no errors are found for that vPE, the search incrementally checks the other vPEs. The search wraps around to ensure all vPEs are searched. The search ends when an error is found or when the search has checked all the vPEs.	RW

When you read the GICR_VERRR register, the following tables show the bit assignments for the different request types:

Response to a Doorbell read request

The following table shows the bit assignments when the GIC performs a read (RD) request of the Doorbell information.

Table 5-78: GICR_VERRR bit assignments, for a Doorbell read request

Bits	Name	Description
[63]	Busy	Indicates if the read request is complete: **0** Doorbell read request is complete. **1** Doorbell read request is in progress.
[62]	Response	Indicates if the request was successful, and is valid only when Busy == 0: **0** The GIC performed the request. **1** The GIC failed to perform the request.
[61:60]	Opcode	Returns 0 because an RD request was requested.
[59]	-	**RES0**
[58]	Errored	Indicates if the request has errored in the vTGT cache: **0** The request did not cause an error. **1** The request has errored in the vTGT cache. The Doorbell ID might be incorrect.
[57:42]	DB_ID	Returns the default Doorbell identifier.
[41]	DB_Mask	Returns the default Doorbell mask.
[40:38]	-	**RES0**
[37]	DB_Prop	Indicates if the default Doorbell properties are valid: **0** The default Doorbell properties are not valid. **1** The default Doorbell properties are valid.
[36]	DB_Enabled	Indicates if the default Doorbell is enabled: **0** The default Doorbell is not enabled. **1** The default Doorbell is enabled.
[35:32]	DB_Priority	Returns the priority of the default Doorbell. `0b0000` is the lowest priority and `0b1111` is the highest priority.
[31:9]	-	**RES0**
[8:0]	DB_PE	Returns the PE that the default Doorbell targets.

Response to a vPT read request

The following table shows the bit assignments when the GIC performs a read (RD) request of the vPT information.

Table 5-79: GICR_VERRR bit assignments, for a vPT read request

Bits	Name	Description
[63]	Busy	Indicates if the read request is complete: **0** vPT read request is complete. **1** vPT read request is in progress.

Bits	Name	Description
[62]	Response	Indicates if the request was successful, and is valid only when Busy == 0: **0** The GIC performed the request. **1** The GIC failed to perform the request.
[61:60]	Opcode	Returns 0 because an RD request was requested.
[59]	Mapped	Indicates if the vPE is mapped on the local chip: **0** The vPE is not mapped on the local chip. **1** The vPE is mapped on the local chip.
[58]	Errored	Indicates if the vPE is errored: **0** The vPE is not errored. **1** The vPE is errored.
[57:42]	Mapped_ITS	Returns the ITSs that the vPE is mapped on: • Bit[57] is ITS15 • Bit[56] is ITS14 • ... • Bit[42] is ITS0
[41:36]	-	**RES0**
[35:0]	vPT_Addr	Returns the vPT base address, bits[51:15], for the vPE.

Response to a vCONF read request

The following table shows the bit assignments when the GIC performs a read (RD) request of the vCONF information.

Table 5-80: GICR_VERRR bit assignments, for a vCONF read request

Bits	Name	Description
[63]	Busy	Indicates if the read request is complete: **0** vCONF read request is complete. **1** vCONF read request is in progress.
[62]	Response	Indicates if the request was successful, and is valid only when Busy == 0: **0** The GIC performed the request. **1** The GIC failed to perform the request.
[61:60]	Opcode	Returns 0 because an RD request was requested
[59]	Mapped	Indicates if the vPE is mapped on the local chip: **0** The vPE is not mapped on the local chip. **1** The vPE is mapped on the local chip.
[58]	Errored	Indicates if the vPE is errored: **0** The vPE is not errored. **1** The vPE is errored.

Bits	Name	Description
[57:42]	Mapped_ITS	When GICD_CFGID.EITS == 1, returns the ITSs that the vPE is mapped on: • Bit[57] is ITS31 • Bit[56] is ITS30 • ... • Bit[42] is ITS16 When GICD_CFGID.EITS == 0, this field is **RES0**.
[41:36]	-	**RES0**
[35:0]	vCONF_Addr	Returns the vCONF base address, bits[51:15], for the vPE.

Response to a vSGI read request

The following table shows the bit assignments when the GIC performs a read (RD) request of the vSGI programming information.

Table 5-81: GICR_VERRR bit assignments, for a vSGI read request

Bits	Name	Description
[63]	Busy	Indicates if the read request is complete: **0** vSGI read request is complete. **1** vSGI read request is in progress.
[62]	Response	Indicates if the request was successful, and is valid only when Busy == 0: **0** The GIC performed the request. **1** The GIC failed to perform the request.
[61:60]	Opcode	Returns 0 because an RD request was requested
[59]	-	**RES0**
[58]	Errored	Indicates if the request has errored in the vTGT cache: **0** The request did not cause an error. **1** The request has errored in the vTGT cache. The vSGI programming might be incorrect.
[57:48]	-	**RES0**
[47:40]	vSGI_Group	Each bit represents a vSGI and it indicates which group the vSGI belongs to: **0** The vSGI belongs to Group 0. **1** The vSGI belongs to Group 1. Bit[40] represents vSGI[0] and bit[47] represents vSGI[7].
[39:32]	vSGI_Enabled	Each bit represents a vSGI and indicates if the vSGI is enabled: **0** The vSGI is not enabled. **1** The vSGI is enabled. Bit[32] represents vSGI[0] and bit[39] represents vSGI[7].
[32:0]	vSGI_Priority	Each nibble represents a vSGI and it returns the priority of the vSGI. `0b0000` is the lowest priority and `0b1111` is the highest priority. Bits[3:0] represent vSGI[0] and bits[31:28] represent vSGI[7].

Accessibility

GICR_VERRR is accessible only with a 64-bit access.

5.7 ITS control register summary

The GIC-720AE *Interrupt Translation Service* (ITS) functions are controlled through registers that are identified with the prefix GITS.

This page does not exist in GIC-720AE configurations that do not support LPIs.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, these GITS registers are accessible for view 0 and view 1.

For descriptions of registers that are not specific to the GIC-720AE, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Table 5-82: ITS control register summary

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0000	GITS_CTLR	RW	0x80000000	32	ITS Control Register	Yes
0x0004	GITS_IIDR	RO	0x070nn43B The nn value depends on the r*x*p*y* identifier.	32	ITS Implementer Identification Register	Yes
0x0008	GITS_TYPER	RO	Configuration dependent	64	ITS Type Register	Yes
0x0010	GITS_MPAMIDR	RO	0x000101FF	32	MPAM ID Register	Yes
0x0014	GITS_PARTIDR	RW	0x0	32	Part ID Register	Yes
0x0018-0x001C	-	-	-	-	Reserved	-
0x0020	GITS_FCTLR	RW	0x0	32	Function Control Register	No
0x0024	-	-	-	-	Reserved	-
0x0028	GITS_OPR	RW	0x0	64	Operations Register	No
0x0030	GITS_OPSR	RO	0x0	64	Operation Status Register	No
0x0038-0x007C	-	-	-	-	Reserved	-
0x0080	GITS_CBASER	RW	0x0	64	Command Queue Control Register. See the *Locality-Specific Peripheral Interrupts, Arm® Generic Interrupt Controller v3 and v4*.	Yes
0x0088	GITS_CWRITER	RW	0x0	64	Command Queue Write Pointer Register	Yes
0x0090	GITS_CREADR	RO	0x0	64	Command Queue Read Pointer Register	Yes
0x0098-0x00FC	-	-	-	-	Reserved	-
0x0100	GITS_BASER0	RW	0x0107000000000000	64	ITS Translation Table Descriptor Register0	Yes
0x0108	GITS_BASER1	RW	0x0401000000000000	64	ITS Translation Table Descriptor Register1	Yes
0x0110	GITS_BASER2	RW	Configuration dependent	64	ITS Translation Table Descriptor Register2	Yes

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0118-0xDFFC	-	-	-	-	Reserved	-
0xC000	GITS_D_ERRINSR	RW	Configuration dependent	64	Device Cache error injection	No
0xC008	GITS_V_ERRINSR	RW	Configuration dependent	64	Event Cache error injection	No
0xC010	GITS_C_ERRINSR	RW	Configuration dependent	64	Collection Cache error injection	No
0xC018-0xEFFC	-	-	-	-	Reserved	-
0xF000	GITS_CFGID	RO	Configuration dependent	64	Configuration ID Register	No
0xF008-0xFFCC	-	-	-	-	Reserved	-
0xFFD0	GITS_PIDR4	RO	0x44	32	Peripheral ID 4 Register	No
0xFFD4	GITS_PIDR5	RO	0x00	32	Peripheral ID 5 Register	No
0xFFD8	GITS_PIDR6	RO	0x00	32	Peripheral ID 6 Register	No
0xFFDC	GITS_PIDR7	RO	0x00	32	Peripheral ID 7 Register	No
0xFFE0	GITS_PIDR0	RO	0x94	32	Peripheral ID 0 Register	No
0xFFE4	GITS_PIDR1	RO	0xB4	32	Peripheral ID 1 Register	No
0xFFE8	GITS_PIDR2	RO	Configuration dependent	32	Peripheral ID 2 Register	No
0xFFEC	GITS_PIDR3	RO	0x00	32	Peripheral ID 3 Register	No
0xFFF0	GITS_CIDR0	RO	0x0D	32	Component ID 0 Register	No
0xFFF4	GITS_CIDR1	RO	0xF0	32	Component ID 1 Register	No
0xFFF8	GITS_CIDR2	RO	0x05	32	Component ID 2 Register	No
0xFFFC	GITS_CIDR3	RO	0xB1	32	Component ID 3 Register	No

5.7.1 GITS_IIDR, ITS Implementer Identification Register

This register provides information about the implementer and revision of the ITS.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 32-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-64: GITS_IIDR bit assignments

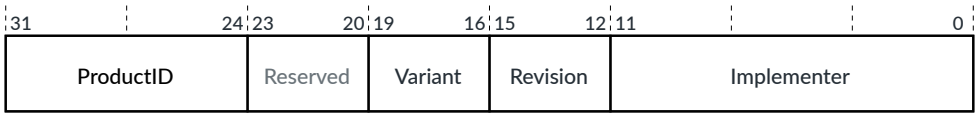


Table 5-83: GITS_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **0x07** GIC-720AE
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: **0x0** r0 **0x1** r1 **0x2** r2
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: **0x0** p0 **0x1** p1
[11:0]	Implementer	Identifies the implementer: **0x43B** Arm

5.7.2 GITS_TYPER, ITS Type Register

This register returns information about the features that this ITS supports.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-65: GITS_TYPER bit assignments

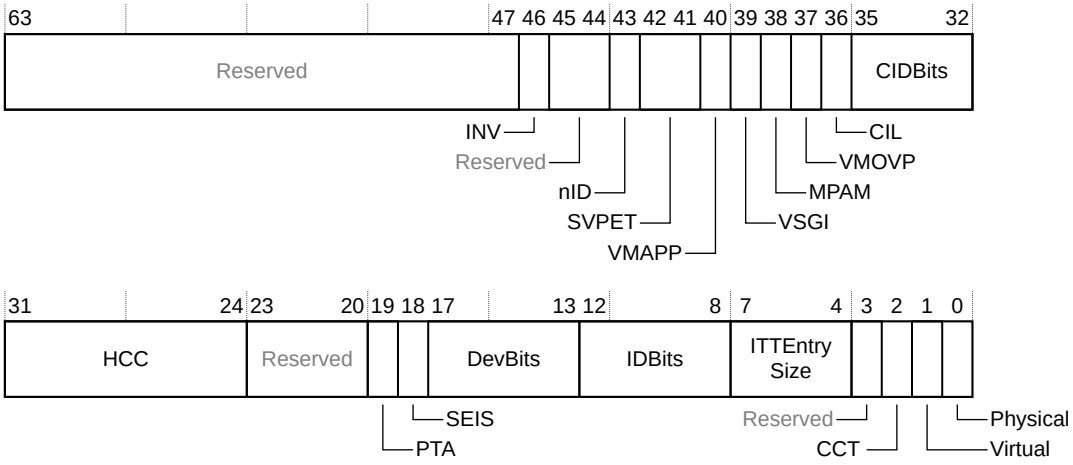


Table 5-84: GITS_TYPER bit descriptions

Bits	Name	Description
[63:47]	-	Reserved, RAZ
[46]	INV	Returns 1, to indicate that: • The Device cache and Event cache are invalidated when writing to GITS_BASER0. • The Collection cache is invalidated when writing to GITS_BASER1.
[45:44]	-	Reserved, RAZ
[43]	nID	Indicates whether GIC-720AE supports individual doorbells: **1** Individual doorbell is not supported.
[42:41]	SVPET	Returns: **0b00** vPE table is not shared with Redistributors. This bit value occurs when the GIC does not support GICv4.1. **0b01** vPE table is shared with the groups of Redistributors that GITS_MPIDR.Aff3 indicates. This bit value occurs for all configurations of the GIC except for a multichip configuration with `chip_affinity_select_level` == 2. **0b10** vPE table is shared with the groups of Redistributors that GITS_MPIDR.[Aff3, Aff2] indicate. This bit value occurs for a GIC multichip configuration with `chip_affinity_select_level` == 2. When this field is not 0, it reports the same value as the GICR_TYPER.CommonLPIAff field of the Redistributors it shares the table with.
[40]	VMAPP	Returns 1, to indicate a GICv4.1 `VMAPP` command layout.
[39]	VSGI	Indicates whether this ITS supports direct injection of SGIs: **0** This ITS does not support direct injection of SGIs. This value occurs when `gicv41_support` == 0. **1** This ITS supports direct injection of SGIs. This value occurs when `gicv41_support` == 1.
[38]	MPAM	Indicates whether the ITS supports *Memory Partitioning and Monitoring* (MPAM): **0** MPAM is not supported. This value occurs when `lpi_support` == 0. **1** MPAM is supported. This value occurs when `lpi_support` == 1.

Bits	Name	Description
[37]	VMOVP	Indicates the form of the `VMOVP` command: **0** This bit value occurs when `gicv41_support` == 0. **1** When software moves a vPE, then it need only issue a `VMOVP` on one of the ITSs that has a mapping for that vPE. The ITSList and Sequence Number fields in the `VMOVP` command are **RES0**. This bit value occurs when `gicv41_support` == 1.
[36]	CIL	Collection ID limit: **1** The size of the Collection ID is set by the CIDBits field.
[35:32]	CIDBits	The number of Collection ID bits, minus one. Set by the `col_width` configuration parameter.
[31:24]	HCC	Hardware collection count: **0** Interrupt collections are held in external memory only.
[23:20]	-	Reserved, returns 0
[19]	PTA	Physical target addresses: **0** The GIC-720AE does not support physical target addresses.
[18]	SEIS	System error interrupts: **0** The GIC-720AE does not support locally generated System Error interrupts.
[17:13]	DevBits	The number of device identifier bits implemented, minus one. Set by the `did_width` configuration parameter.
[12:8]	IDBits	The number of interrupt identifier bits implemented, minus one. Set by the `vid_width` configuration parameter.
[7:4]	ITTEntrySize	The number of bytes for each entry, minus one: **0x3** The GIC-720AE supports a 4-byte ITT entry size.
[3]	-	Reserved
[2]	CCT	Cumulative Collection tables: **0** Total number of supported collections is determined by the number of collections that are held in memory only.
[1]	Virtual	Indicates whether the ITS supports virtual LPIs and direct injection of virtual LPIs: **0** The ITS does not support virtual LPIs or direct injection of virtual LPIs. This bit value occurs when `gicv41_support` == 0. **1** The ITS supports virtual LPIs and direct injection of virtual LPIs. This bit value occurs when `gicv41_support` == 1. See the *Arm® Generic Interrupt Controller v3 and v4 - Virtualization*.
[0]	Physical	Physical LPIs: **1** The GIC-720AE supports physical LPIs.

5.7.3 GITS_MPAMIDR, MPAM ID Register

This register returns the maximum values that the *Memory Partitioning and Monitoring* (MPAM) fields can be set to in GITS_PARTIDR.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 32-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-66: GITS_MPAMIDR bit assignments

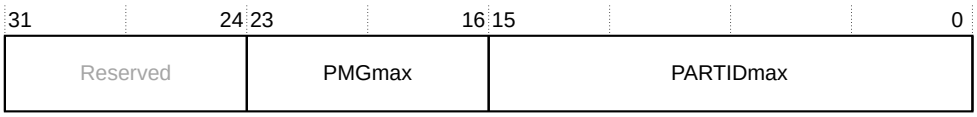


Table 5-85: GITS_MPAMIDR bit descriptions

Bits	Name	Description
[31:24]	-	Reserved
[23:16]	PMGmax	Performance monitoring group. Returns `0x01`, and indicates the maximum value that GITS_PARTIDR.PMG can be set to.
[15:0]	PARTIDmax	Returns `0x01FF`, and indicates the maximum value that GITS_PARTIDR.PARTID can be set to.

5.7.4 GITS_PARTIDR, PART ID Register

This register sets the Partition ID and PMG values that the ITS uses during memory accesses.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 32-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-67: GITS_PARTIDR bit assignments

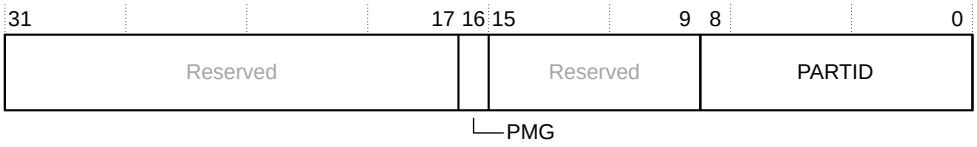


Table 5-86: GITS_PARTIDR bit descriptions

Bits	Name	Description
[31:17]	-	Reserved
[16]	PMG	The performance monitoring group value that the ITS uses when it accesses memory
[15:9]	-	Reserved
[8:0]	PARTID	The Partition ID value that the ITS uses when it accesses memory

5.7.5 GITS_FCTLR, Function Control Register

This register controls many functions in the local GITS such as cache invalidation, clock gating, and the scrubbing of all RAMs. The register is not distributed and only acts on the local chip.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 32-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

If the ITS is not quiescent, then the GIC ignores writes to some fields. The ITS is quiescent when GITS_CTLR.Quiescent == 1.

Bit descriptions

Figure 5-68: GITS_FCTLR bit assignments

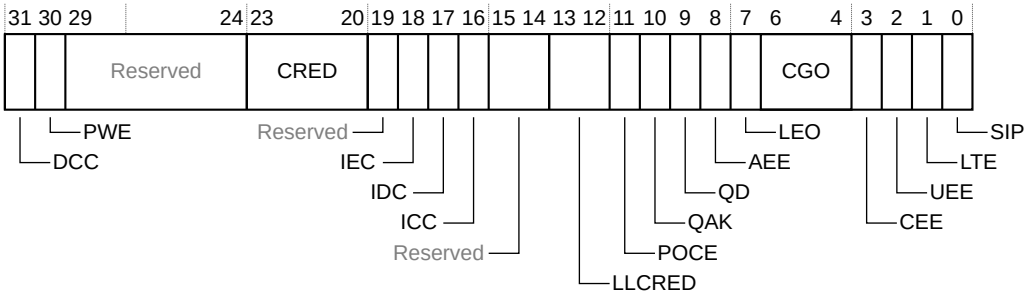


Table 5-87: GITS_FCTLR bit descriptions

Bits	Name	Description	Type
[31]	DCC	Disable cache conversion: **0** Use SMMU attribute for AMBA mapping. **1** Use Direct attribute for AMBA mapping. Writes ignored if the ITS is not quiescent.	RW
[30]	PWE	Powerdown when enabled: **0** Requests GITS_CTLR.Quiescent to indicate that the ITS is quiescent and can be powered down. **1** Do not request GITS_CTLR.Quiescent to indicate that the ITS is quiescent.	RW
[29:24]	-	Reserved, RAZ/WI	-
[23:20]	CRED	LPI credit initialization: **0x0** Default to the configured credit value of GITS_CFGID.LPI_Credit_Count + 1. **0x1** 1 credit **0x2** 2 credits **...** ... **0xE** 14 credits **0xF** 15 credits	RW
[19]	-	Reserved, RAZ/WI	-
[18]	IEC	Invalidate Event cache: When written: **0** No effect. **1** Invalidate Event cache. When read: **0** Invalidation complete. **1** Event cache invalidation in progress, including the BASER0 write-initiated invalidate.	RW

Bits	Name	Description	Type
[17]	IDC	Invalidate Device cache: When written: **0** No effect. **1** Invalidate Device cache. When read: **0** Invalidation complete. **1** Device cache invalidation in progress, including the BASER0 write-initiated invalidate.	RW
[16]	ICC	Invalidate Collection cache: When written: **0** No effect. **1** Invalidate Collection cache. When read: **0** Invalidation complete. **1** Collection cache invalidation in progress, including the BASER1 write-initiated invalidate.	RW
[15:14]	-	Reserved, RAZ/WI	-
[13:12]	LLCRED	Low-latency LPI credit: **0b00** Default to the configured credit value of GITS_CFGID.Low_Latency_LPI_Credit_Count. **0b01** 1 credit **0b10** 2 credits **0b11** 3 credits	RW
[11]	POCE	Poison check enable: **0** Disable poison checking on the ACE5-Lite subordinate port. **1** Enable poison checking on the ACE5-Lite subordinate port.	RW
[10]	QAK	Quiescent ACK override: **0** Disable quiescent ACK override. **1** Enable quiescent ACK override.	RW
[9]	QD	Q-Channel deny: **0** Do not deny Q-Channel requests. **1** Always deny Q-Channel requests.	RW
[8]	AEE	Access error enable: **0** Do not enable reporting of subordinate access errors. **1** Enable reporting of subordinate access errors. Writes ignored if the ITS is not quiescent.	RW
[7]	LEO	LPI error overflow. **0** LPI errors are always sent. **1** To prevent excessive debug messages, LPI errors set the overflow bit in debug messages. Writes ignored if the ITS is not quiescent.	RW

Bits	Name	Description	Type
[6:4]	CGO	Clock gate override. One bit for each clock gate: **0** Use full clock gating. **1** Leave clock running. If clock gates are not implemented, then you must use this value. The clock gate bit assignments are: **Bit[6], CGO[2]** Debug clock **Bit[5], CGO[1]** Command clock **Bit[4], CGO[0]** ITU clock	RW
[3]	CEE	Command error enable: **0** Do not enable reporting of command errors and errors from GITS_OPR operations. **1** Enable reporting of command errors and errors from GITS_OPR operations. See 4.17.4.15 ITS command and translation error records 27+ on page 114. Writes ignored if the ITS is not quiescent.	RW
[2]	UEE	Unmapped error enable: **0** Do not enable reporting of unmapped interrupt errors. **1** Enable reporting of unmapped interrupt errors. Writes ignored if the ITS is not quiescent.	RW
[1]	LTE	Latency tracking enable: **0** Disable latency tracking of interrupts. **1** Enable latency tracking of interrupts. Writes ignored if the ITS is not quiescent.	RW
[0]	SIP	Scrub in progress. When read: **0** No scrub in progress. **1** Scrub in progress. When written: **0** Abort the scrub. **1** Start a scrub. When a scrub is complete, the GIC clears the bit to 0.	RW

5.7.6 GITS_OPR, Operations Register

This register controls cache lock.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-69: GITS_OPR bit assignments

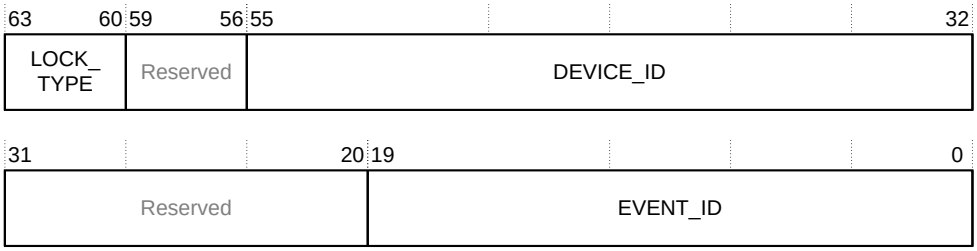


Table 5-88: GITS_OPR bit descriptions

Bits	Name	Description
[63:60]	LOCK_TYPE	Lock type supported: **0** Track **1** Trial **2** ITS lock **3** ITS unlock **4** Track abort **8** ITS unlock all **5-7, 9-15** Reserved **Note:** • If GITS_OPSR.REQUEST_IN_PROGRESS == 1 and software attempts a new access (other than Track abort (4) during a Track), then the behavior is unpredictable. • Invalidating the Event cache by using GITS_FCTLR.IEC unlocks all the locked entries. However, if a GITS_OPR lock request occurs while an invalidation is in progress (GITS_FCTLR.IEC == 1), then it is unpredictable whether the entries remain locked when the invalidation completes. This unpredictable behavior might cause GITS_OPSR to return an incorrect status.
[59:56]	-	Reserved, RES0
[55:32]	DEVICE_ID	Sets the DeviceID. The number of bits that are implemented in this field is configuration dependent. To determine the width of this field, software can read GITS_TYPER.DevBits.
[31:20]	-	Reserved, RES0
[19:0]	EVENT_ID	Sets the EventID. The number of bits that are implemented in this field is configuration dependent. To determine the width of this field, software can read GITS_TYPER.IDBits.

5.7.7 GITS_OPSR, Operation Status Register

This register indicates cache lock status.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-70: GITS_OPSR bit assignments

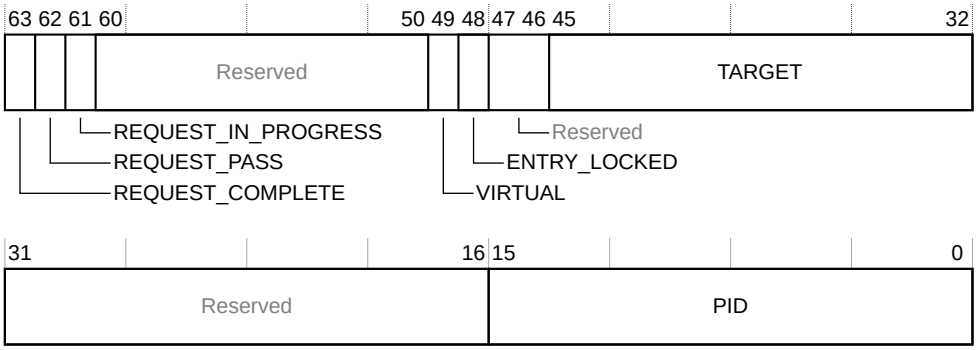


Table 5-89: GITS_OPSR bit descriptions

Bits	Name	Description
[63]	REQUEST_COMPLETE	Request to GITS_OPR completed
[62]	REQUEST_PASS	Request to GITS_OPR completed without error
[61]	REQUEST_IN_PROGRESS	Request to GITS_OPR in progress
[60:50]	-	Reserved, **RES0**
[49]	VIRTUAL	Indicates whether the interrupt is virtual or physical: **0** A physical interrupt is targeting the PE that GITS_OPSR.TARGET selects **1** A virtual interrupt is targeting the vPE that GITS_OPSR.TARGET selects Valid for trial and lock operations.
[48]	ENTRY_LOCKED	Locked entry in cache corresponds to request (valid for trial and lock operations)
[47:46]	-	Reserved, **RES0**

Bits	Name	Description
[45:32]	TARGET	Target of interrupt, which is either: • a vPE when GITS_OPSR.VIRTUAL == 1 • a PE when GITS_OPSR.VIRTUAL == 0 Valid for trial and lock operations.
[31:16]	-	Reserved, **RES0**
[15:0]	PID	ID of interrupt requested (valid for trial and lock operations)

5.7.8 GITS_D_ERRINSR, Error Insertion Device cache register

This register can insert errors into the ITS Device cache RAM. You can use this register to test your error recovery software.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

The bit assignments within this register depend on whether a write access or read access occurs.

The following table shows the bit assignments for write accesses.

Table 5-90: GITS_D_ERRINSR bit assignments for writes

Bits	Name	Description
[63]	Valid	Set to 1, to start the error injection process. The GIC sets this bit to 0 when it completes the process.
[62:61]	-	**RES0**
[60]	DisableWriteCheck	Controls whether to include an encoding check: **0** Include an encoder check. **1** Disable an encoder check.
[59:48]	-	**RES0**
[47:32]	ADDR	Address

Bits	Name	Description
[31]	ERRINS2VALID	Controls whether the second error is valid: **0** The ERRINS2LOC field is not valid. **1** The ERRINS2LOC field is valid.
[30:25]	-	**RES0**
[24:16]	ERRINS2LOC	Sets the address location of the second error.
[15]	ERRINS1VALID	Controls whether the first error is valid: **0** The ERRINS1LOC field is not valid. **1** The ERRINS1LOC field is valid.
[14:9]	-	**RES0**
[8:0]	ERRINS1LOC	Sets the address location of the first error.

The following table shows the bit assignments for read accesses.

Table 5-91: GITS_D_ERRINSR bit assignments for reads

Bits	Name	Description
[63]	Valid	Indicates if the error injection process is complete: **0** Error injection process is complete. **1** Error injection process is in progress.
[62:61]	Status	Indicates if the error injection process was successful, and is valid only when Valid == 0: **0b00** The GIC performed the error injection process. **0b01** An out-of-range error occurred. To fix this error, check that the RAM ID and the error locations are correct. **0b10** A coincident error occurred. **0b11** An encoder or decoder mismatch occurred.
[60]	RAM_Present	Indicates whether a RAM with ECC is present: **1** RAM with ECC is present.
[59:48]	-	**RES0**
[47:32]	RAM_MAX	Returns the maximum address of the RAM.
[31:9]	-	**RES0**
[8:0]	RAM WIDTH	Returns the highest maximum bit width of the RAM. For example, a value of 15 indicates a 16-bit wide RAM.

Accessibility

If GICD_SAC.GICTNS == 0, then GITS_D_ERRINSR is accessible only by Secure accesses.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, GITS_D_ERRINSR is accessible only for view 0.

5.7.9 GITS_V_ERRINSR, Error Insertion Event cache register

This register can insert errors into the ITS Event cache RAM. You can use this register to test your error recovery software.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

The bit assignments within this register depend on whether a write access or read access occurs.

The following table shows the bit assignments for write accesses.

Table 5-92: GITS_V_ERRINSR bit assignments for writes

Bits	Name	Description
[63]	Valid	Set to 1, to start the error injection process. The GIC sets this bit to 0 when it completes the process.
[62:61]	-	**RES0**
[60]	DisableWriteCheck	Controls whether to include an encoding check: **0** Include an encoder check. **1** Disable an encoder check.
[59:48]	-	**RES0**
[47:32]	ADDR	Address
[31]	ERRINS2VALID	Controls whether the second error is valid: **0** The ERRINS2LOC field is not valid. **1** The ERRINS2LOC field is valid.
[30:25]	-	**RES0**
[24:16]	ERRINS2LOC	Sets the address location of the second error.
[15]	ERRINS1VALID	Controls whether the first error is valid: **0** The ERRINS1LOC field is not valid. **1** The ERRINS1LOC field is valid.
[14:9]	-	**RES0**
[8:0]	ERRINS1LOC	Sets the address location of the first error.

The following table shows the bit assignments for read accesses.

Table 5-93: GITS_V_ERRINSR bit assignments for reads

Bits	Name	Description
[63]	Valid	Indicates if the error injection process is complete: **0** Error injection process is complete. **1** Error injection process is in progress.
[62:61]	Status	Indicates if the error injection process was successful, and is valid only when Valid == 0: **0b00** The GIC performed the error injection process. **0b01** An out-of-range error occurred. To fix this error, check that the RAM ID and the error locations are correct. **0b10** A coincident error occurred. **0b11** An encoder or decoder mismatch occurred.
[60]	RAM_Present	Indicates whether a RAM with ECC is present: **1** RAM with ECC is present.
[59:48]	-	**RES0**
[47:32]	RAM_MAX	Returns the maximum address of the RAM.
[31:9]	-	**RES0**
[8:0]	RAM WIDTH	Returns the highest maximum bit width of the RAM. For example, a value of 15 indicates a 16-bit wide RAM.

Accessibility

If GICD_SAC.GICTNS == 0, then GITS_V_ERRINSR is accessible only by Secure accesses.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, GITS_V_ERRINSR is accessible only for view 0.

5.7.10 GITS_C_ERRINSR, Error Insertion Collection cache register

This register can insert errors into the ITS Collection cache RAM. You can use this register to test your error recovery software.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

The bit assignments within this register depend on whether a write access or read access occurs.

The following table shows the bit assignments for write accesses.

Table 5-94: GITS_C_ERRINSR bit assignments for writes

Bits	Name	Description
[63]	Valid	Set to 1, to start the error injection process. The GIC sets this bit to 0 when it completes the process.
[62:61]	-	**RES0**
[60]	DisableWriteCheck	Controls whether to include an encoding check: **0** Include an encoder check. **1** Disable an encoder check.
[59:48]	-	**RES0**
[47:32]	ADDR	Address
[31]	ERRINS2VALID	Controls whether the second error is valid: **0** The ERRINS2LOC field is not valid. **1** The ERRINS2LOC field is valid.
[30:25]	-	**RES0**
[24:16]	ERRINS2LOC	Sets the address location of the second error.
[15]	ERRINS1VALID	Controls whether the first error is valid: **0** The ERRINS1LOC field is not valid. **1** The ERRINS1LOC field is valid.
[14:9]	-	**RES0**
[8:0]	ERRINS1LOC	Sets the address location of the first error.

The following table shows the bit assignments for read accesses.

Table 5-95: GITS_C_ERRINSR bit assignments for reads

Bits	Name	Description
[63]	Valid	Indicates if the error injection process is complete: **0** Error injection process is complete. **1** Error injection process is in progress.
[62:61]	Status	Indicates if the error injection process was successful, and is valid only when Valid == 0: **0b00** The GIC performed the error injection process. **0b01** An out-of-range error occurred. To fix this error, check that the RAM ID and the error locations are correct. **0b10** A coincident error occurred. **0b11** An encoder or decoder mismatch occurred.
[60]	RAM_Present	Indicates whether a RAM with ECC is present: **1** RAM with ECC is present.
[59:48]	-	**RES0**

Bits	Name	Description
[47:32]	RAM_MAX	Returns the maximum address of the RAM.
[31:9]	-	**RES0**
[8:0]	RAM WIDTH	Returns the highest maximum bit width of the RAM. For example, a value of 15 indicates a 16-bit wide RAM.

Accessibility

If GICD_SAC.GICTNS == 0, then GITS_C_ERRINSR is accessible only by Secure accesses.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, GITS_C_ERRINSR is accessible only for view 0.

5.7.11 GITS_CFGID, Configuration ID Register

This register returns information about the configuration of the ITS block such as its ID number.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-71: GITS_CFGID bit assignments

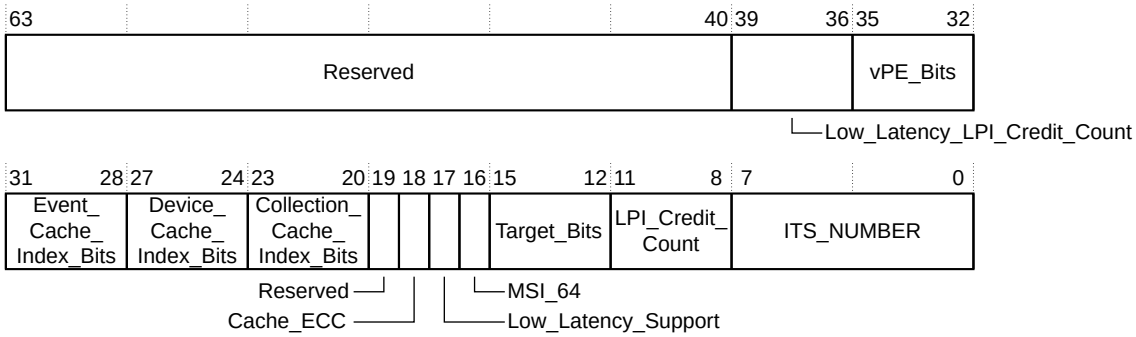


Table 5-96: GITS_CFGID bit descriptions

Bits	Name	Description
[63:40]	-	Reserved, RES0

Bits	Name	Description
[39:36]	Low_Latency_LPI_Credit_Count	Number of low-latency LPI credits. The `number_ll_int_credit` configuration parameter sets the value of this field.
[35:32]	vPE_Bits	Number of bits that are used for vPE IDs.
[31:28]	Event_Cache_Index_Bits	Number of bits that are used to index the Event cache.
[27:24]	Device_Cache_Index_Bits	Number of bits that are used to index the Device cache.
[23:20]	Collection_Cache_Index_Bits	Number of bits that are used to index the Collection cache.
[19]	-	Reserved
[18]	Cache_ECC	Translation caching has ECC protection.
[17]	Low_Latency_Support	Lock translations in cache support.
[16]	MSI_64	MSI-64 Encapsulator support. The `msi_64` configuration parameter sets the value of this bit.
[15:12]	Target_Bits	Number of bits supported for targets.
[11:8]	LPI_Credit_Count	Number of LPI credits – 1. The `number_int_credit` configuration parameter minus 1, sets the value of this field.
[7:0]	ITS_Number	Returns the ITS block ID. The its_id[7:0] tie-off signal controls the ID value. Each ITS block must have a unique ID.

5.7.12 GITS_PIDR2, Peripheral ID2 Register

This register returns byte[2] of the peripheral ID. The GITS_PIDR2 register is part of the set of ITS peripheral identification registers.

Configurations

This register is available in all configurations that have one or more ITS blocks.

Attributes

Width 32-bit
Functional group See 5.7 ITS control register summary on page 229 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-72: GITS_PIDR2 bit assignments

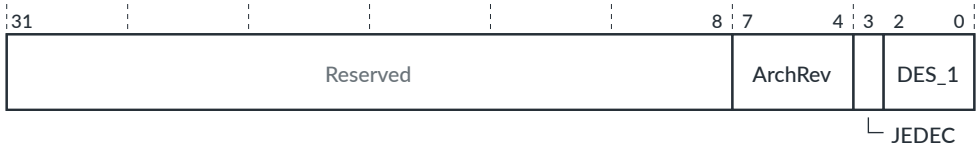


Table 5-97: GITS_PIDR2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:4]	ArchRev	Identifies the version of the GIC architecture with which the ITS complies: **0x3** GICv3 **0x4** GICv4
[3]	JEDEC	Indicates that a JEDEC-assigned JEP106 identity code is used
[2:0]	DES_1	Bits[6:4] of the JEP106 identity code. Bits[3:0] of the JEP106 identity code are assigned to GITS_PIDR1[7:4].

5.8 ITS translation register summary

Interrupts to be translated by the GIC-720AE *Interrupt Translation Service* (ITS) are identified by EventIDs that are written to GITS_TRANSLATER, the ITS Translation Register.

This page does not exist in GIC-720AE configurations that do not support LPIs or that do not have an ITS.

Table 5-98: ITS translation register summary

Offset	Name	Type	Reset	Width	Description
0x0000-0x003C	-	-	-	-	Reserved
0x0040	GITS_TRANSLATER	WO	-	32	ITS Translation Register. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*. For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, the GITS_TRANSLATER ignores the value of view.
0x0044-0xFFFC	-	-	-	-	Reserved

5.9 ITS vSGI register summary

Virtual SGIs to be injected directly into a virtual machine are written to the ITS translation register GITS_SGIR.

This page does not exist in GIC-720AE configurations that do not support vSGIs or that do not have an ITS. For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, the GITS_SGIR register is accessible for view 0 and view 1 only.

Table 5-99: ITS vSGI register summary

Offset	Name	Type	Reset	Width	Description
0x0000-0x001C	-	-	-	-	Reserved

Offset	Name	Type	Reset	Width	Description
0x0020	GITS_SGIR	WO	-	64	ITS vSGI Register. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.
0x0028-0xFFFC	-	-	-	-	Reserved

5.10 GICT register summary

The GIC-720AE trace and debug functions are controlled through registers that are identified with the prefix GICT.

All registers comply with the *RAS System Architecture* chapter of the *Arm® Architecture Reference Manual for A-profile architecture*, except for the GICT_PIDR* and GICT_CIDR* registers.

Note

The GICD_SAC.GICTNS bit controls whether Non-secure software can access the GICT registers.

Table 5-100: GICT register summary

Offset	Name	Type	Reset	Width	Description
0x0000 + (n × 64)	GICT_ERR<n>FR	RO	Record dependent	64	Error Record Feature Register
0x0008 + (n × 64)	GICT_ERR<n>CTLR	RW	0x0	64	Error Record Control Register
0x0010 + (n × 64)	GICT_ERR<n>STATUS	RW	Record dependent	64	Error Record Primary Status register
0x0018 + (n × 64)	GICT_ERR<n>ADDR	RW	Unknown	64	Error Record Address Register
0x0020 + (n × 64)	GICT_ERR<n>MISC0	RW	Unknown	64	Error Record Miscellaneous Register 0
0x0028 + (n × 64)	GICT_ERR<n>MISC1	RW	Unknown	64	Error Record Miscellaneous Register 1
0xE000	GICT_ERRGSR	RO	0x0	64	Error Group Status Register
0xE008-0xE0FC	-	-	-	-	Reserved, RAZ/WI
0xE100	GICT_IIDR	RO	0x070nn43B The nn value depends on the r*x*p*y* identifier.	32	Trace Implementer Identification Register
0xE104-0xE7FC	-	-	-	-	Reserved, RAZ/WI
0xE800-0xE808	GICT_ERRIRQCR<n>	RW	0x0	64	Error Interrupt Configuration Registers
0xE810-0xFFB8	-	-	-	-	Reserved, RAZ/WI

Offset	Name	Type	Reset	Width	Description
0xFFBC	GICT_DEVARCH	RO	0x47700A00	32	Device Architecture register
0xFFC0-0xFFC4	-	-	-	-	Reserved, RAZ/WI
0xFFC8	GICT_DEVID	RO	Configuration dependent	32	Device Configuration register
0xFFCC	-	-	-	-	Reserved, RAZ/WI
0xFFD0	GICT_PIDR4	RO	0x44	32	Peripheral ID 4 register
0xFFD4	GICT_PIDR5	RO	0x00	32	Peripheral ID 5 register
0xFFD8	GICT_PIDR6	RO	0x00	32	Peripheral ID 6 register
0xFFDC	GICT_PIDR7	RO	0x00	32	Peripheral ID 7 register
0xFFE0	GICT_PIDR0	RO	0x95	32	Peripheral ID 0 register
0xFFE4	GICT_PIDR1	RO	0xB4	32	Peripheral ID 1 register
0xFFE8	GICT_PIDR2	RO	0x3B	32	Peripheral ID 2 register
0xFFEC	GICT_PIDR3	RO	0x00	32	Peripheral ID 3 register
0xFFF0	GICT_CIDR0	RO	0x0D	32	Component ID 0 register
0xFFF4	GICT_CIDR1	RO	0xF0	32	Component ID 1 register
0xFFF8	GICT_CIDR2	RO	0x05	32	Component ID 2 register
0xFFFC	GICT_CIDR3	RO	0xB1	32	Component ID 3 register

5.10.1 GICT_ERR<n>FR, Error Record Feature Register

This register returns information about the Armv8.2 RAS features that the GIC-720AE implements.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-73: GICT_ERR<n>FR bit assignments

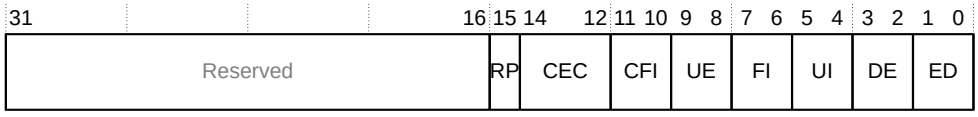


Table 5-101: GICT_ERR<n>FR bit descriptions

Bits	Name	Description
[31:16]	-	Reserved, RAZ
[15]	RP	Repeat corrected error count: **0** The GIC-720AE does not implement a repeat corrected error counter.
[14:12]	CEC	Corrected error count: **0b000** The GIC-720AE does not implement a standard corrected error counter in GICT_ERR<n>MISC0.
[11:10]	CFI	Corrected errors fault interrupt. Depending on the configuration, returns either: **0b00** The GIC-720AE does not provide a fault handling interrupt for corrected errors. **0b10** The GIC-720AE provides a controllable fault handling interrupt for corrected errors.
[9:8]	UE	Uncorrected error. Depending on the configuration, returns either: **0b00** The GIC-720AE does not provide an in-band uncorrected error reporting. **0b10** The GIC-720AE provides a controllable in-band uncorrected error reporting.
[7:6]	FI	Fault handling interrupt for uncorrected errors. Depending on the configuration, returns either: **0b00** The GIC-720AE does not provide a fault handling interrupt. **0b10** The GIC-720AE provides a controllable fault handling interrupt.
[5:4]	UI	Error recovery interrupt for uncorrected errors. Depending on the configuration, returns either: **0b00** The GIC-720AE does not provide an error recovery interrupt for uncorrected errors. **0b10** The GIC-720AE provides a controllable error recovery interrupt for uncorrected errors.
[3:2]	DE	Deferring of errors support: **0b00** The GIC-720AE does not support the deferring of errors.
[1:0]	ED	Uncorrected error reporting: **0b01** Uncorrected error reporting is always enabled.

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_ERR<n>FR is accessible only by Secure accesses.

5.10.2 GICT_ERR<n>CTLR, Error Record Control Register

This register controls how interrupts are handled.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-74: GICT_ERR<n>CTLR bit assignments

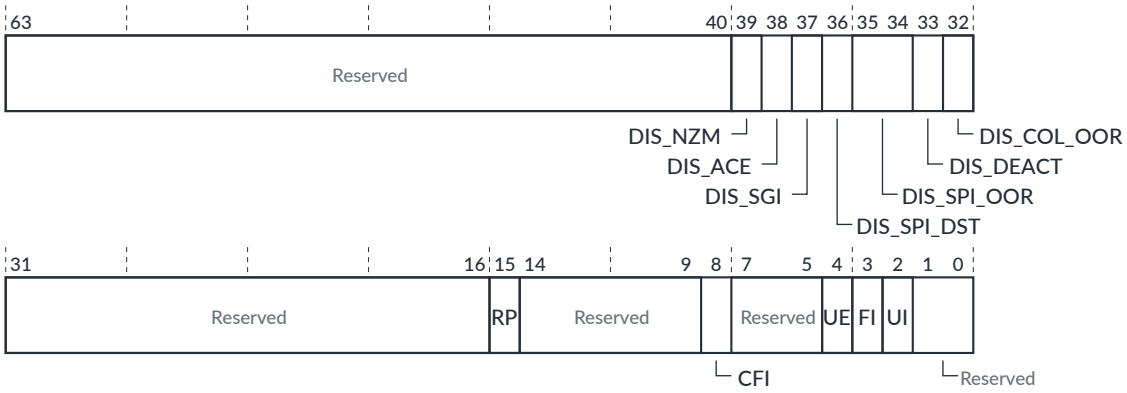


Table 5-102: GICT_ERR<n>CTLR bit descriptions

Bits	Name	Description
[63:40]	-	Reserved, RAZ
[39]	DIS_NZM	This bit can disable the reporting of errors in views 1-3: **0** Reporting of errors occurs in all views, that is, views 0, 1, 2, and 3. **1** Reporting of errors occurs in view 0 only.
[38]	DIS_ACE	RAZ/WI for all records except GICD error record 0. For GICD error record 0, this bit can disable the reporting of illegal ACE accesses: **0** Illegal ACE accesses are treated as errors, which generate the SYN_ACE_BAD syndrome. **1** Reporting of illegal ACE accesses is disabled.
[37]	DIS_SGI	RAZ/WI for all records except GICD error record 0. For GICD error record 0, this bit can disable the reporting of SGIs that are sent with no valid destinations: **0** Out-of-range SGI destinations are treated as errors, which generate the SYN_SGI_NO_TGT syndrome. **1** Reporting of out-of-range SGI destinations is disabled.
[36]	DIS_SPI_DST	RAZ/WI for all records except GICD error record 0. For GICD error record 0, this bit can disable the reporting of SPI destination errors: **0** SPIs with no available destination are treated as errors, which generate either a SYN_SPI_NO_DEST_1OFN or SYN_SPI_NO_DEST_TGT syndrome. **1** Reporting of SPIs with no available destination is disabled.
[35:34]	DIS_SPI_OOR	RAZ/WI for all records except GICD error record 0. For GICD error record 0, this field can disable the reporting of accesses to out-of-range SPIs: **0b00** SPI register accesses to nonexisting blocks are treated as errors, which generate either a SYN_SPI_BLOCK or SYN_SPI_OOR syndrome. **0b01** Reporting of SPI register accesses to all nonexisting blocks is disabled. **0b10** Reporting of SPI register accesses to SPIs 992-1023 is disabled.

Bits	Name	Description
[33]	DIS_DEACT	RAZ/WI for all records except GICD error record 0. For GICD error record 0, this bit can disable the reporting of deactivations to nonexistent SPIs: **0** Out-of-range deactivate messages are treated as errors, which generate the SYN_DEACT_IN syndrome. **1** Reporting of out-of-range deactivate messages is disabled.
[32]	DIS_COL_OOR	RAZ/WI for all records except GICD error record 0. For GICD error record 0, this bit can disable the reporting of an SPI Collator message for a non-implemented SPI: **0** Out-of-range wired SPIs are treated as errors, which generate the SYN_COL_OOR syndrome. **1** Reporting of out-of-range wired SPIs is disabled.
[31:16]	-	Reserved, RAZ
[15]	RP	0 = An error response to a transaction is reported.
[14:9]	-	Reserved, RAZ
[8]	CFI	Controls whether a corrected error generates a fault handling interrupt. SBZ on non-correctable errors else: **0** The GIC-720AE does not assert a fault handling interrupt for corrected errors. **1** The GIC-720AE asserts a fault handling interrupt, the fault_int signal, when a corrected error occurs.
[7:5]	-	Reserved, RAZ
[4]	UE	Uncorrected error. RAZ/WI for all records except GICT error record (0) else: **0** Do not send External abort with transaction. **1** Send External abort with transaction. See 4.17.5 Bus errors on page 130.
[3]	FI	Fault handling interrupt. SBZ on *Correctable Error* (CE) records else: **0** Fault handling interrupt is not generated on any error. **1** Fault handling interrupt, fault_int signal, is generated on all uncorrectable errors.
[2]	UI	Error recovery interrupt for uncorrected error. SBZ on CE records else: **0** Error recovery interrupt is not generated on any error. **1** Error recovery interrupt, err_int signal, is generated on all uncorrectable errors.
[1:0]	-	Reserved, RAZ

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_ERR<n>CTLR is accessible only by Secure accesses.

5.10.3 GICT_ERR<n>STATUS, Error Record Primary Status Register

This register indicates information relating to the recorded errors.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-75: GICT_ERR<n>STATUS bit assignments

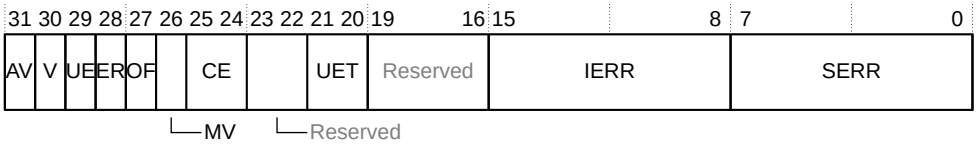


Table 5-103: GICT_ERR<n>STATUS bit descriptions

Bits	Name	Description
[31]	AV	Indicates if the address is valid: **0** GICT_ERR<n>ADDR is not valid. **1** GICT_ERR<n>ADDR contains an address that is associated with the highest priority error that this record stores. Present only in record 0.
[30]	V	Indicates if this register is valid: **0** GICT_ERR<n>STATUS is not valid. **1** GICT_ERR<n>STATUS is valid. One or more errors are recorded.
[29]	UE	Uncorrectable error bit. SBZ in *Correctable Error* (CE) records.
[28]	ER	Indicates that at least one error has been reported over ACE5-Lite. Set for record 0 only, and for accesses only to corrupted data, and bad incoming access.
[27]	OF	Indicates whether multiple errors have been detected. This field is set to 1 when either: • The GICT_ERR<n>MISC0.Count field has overflowed, for records that track correctable ECC errors. • GICT_ERR<n>STATUS.V was previously 1, and a type of error other than a correctable error is recorded.
[26]	MV	Indicates if the GICT miscellaneous registers are valid: **0** GICT_ERR<n>MISC0 and GICT_ERR<n>MISC1 are not valid. **1** GICT_ERR<n>MISC0 and GICT_ERR<n>MISC1 are valid.
[25:24]	CE	Correctable error. Indicates errors that are correctable as shown in Table 4-6: Error handling records on page 98: **0b00** No CE recorded. **0b10** At least one CE recorded.
[23:22]	-	Reserved, RAZ/WI

Bits	Name	Description
[21:20]	UET	Uncorrectable error type. RES0 unless UE == 1, in which case: **0b10** UEO, uncorrectable error and restartable. **0b11** UER, uncorrectable error and recoverable.
[19:16]	-	Reserved, RAZ/WI
[15:8]	IERR	Implementation-defined error code. Returns information that Table 5-106: GICT_ERR<n>MISC0.Data field encoding on page 258 shows. This field is RO apart from record 0 and record 27 (and above).
[7:0]	SERR	Architecturally defined primary error code. Returns information that Table 5-106: GICT_ERR<n>MISC0.Data field encoding on page 258 shows. See the *RAS System Architecture* chapter in the *Arm® Architecture Reference Manual for A-profile architecture* for more information about this field. This field is RO apart from record 0.

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_ERR<n>STATUS is accessible only by Secure accesses.

5.10.4 GICT_ERR<n>ADDR, Error Record Address Register

This register contains the address and security status of the write. This register is present only for GICT software record 0.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

Ignores writes if GICT_ERR<n>STATUS.AV == 1.

All bits are RAZ/WI except when GICT_ERR<n>STATUS.IERR = 0, `0x12`, `0x13`, or `0x14`.

Bit descriptions

Figure 5-76: GICT_ERR<n>ADDR bit assignments

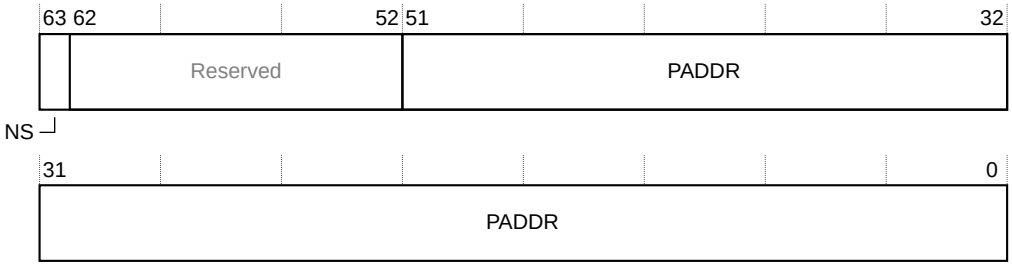


Table 5-104: GICT_ERR<n>ADDR bit descriptions

Bits	Name	Description
[63]	NS	Non-secure attribute: **0** The address is Secure. **1** The address is Non-secure.
[62:52]	-	Reserved, RAZ/WI
[51:0]	PADDR	The error address. The `axis_addr_width` configuration parameter controls how many bits in this field are implemented, that is, from bit[0]-bit[`axis_addr_width`−1].

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_ERR<n>ADDR is accessible only by Secure accesses.

5.10.5 GICT_ERR<n>MISC0, Error Record Miscellaneous Register 0

This register contains the corrected error counter and information that assists with identifying the RAM in which the error was detected.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

None

Bit descriptions

Figure 5-77: GICT_ERR<n>MISC0 bit assignments

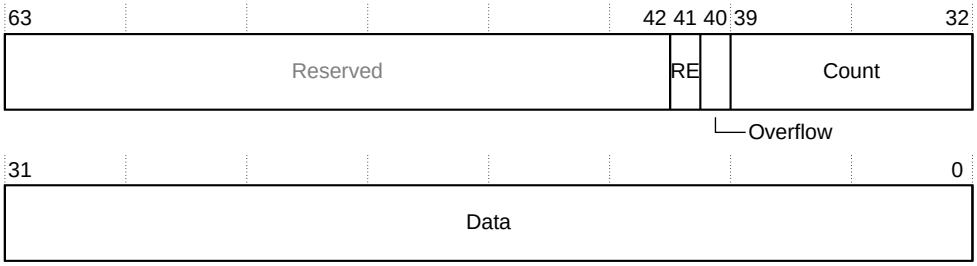


Table 5-105: GICT_ERR<n>MISC0 bit descriptions

Bits	Name	Description
[63:42]	-	Reserved, RAZ
[41]	RE	Rounding error. The rounding error counter is under-reporting.
[40]	Overflow	Sticky overflow bit: **0** Counter has not overflowed. **1** Counter has overflowed. If the corrected fault handling interrupt is enabled, then the GIC-720AE generates a fault handling interrupt.
[39:32]	Count	Error count. Is present for all error records containing RAM errors. Incremented for each corrected error or uncorrectable error that does not match the recorded syndrome.
[31:0]	Data	Information that is associated with the error. A description of each error code is given in one of the following tables: • Table 4-7: Software errors, record 0 on page 100 • Table 4-8: SPI RAM errors, records 1-2 on page 106 • Table 4-9: SGI RAM errors, records 3-4 on page 107 • Table 4-10: TGT_SPI RAM errors, records 5-6 on page 107 • Table 4-11: PPI RAM errors, records 7-8 on page 108 • Table 4-12: LPI RAM errors, records 9-10 on page 109 • Table 4-13: PTS RAM errors, records 11-12 on page 109 • Table 4-14: TGT_LPI RAM errors, records 13-14 on page 110 • Table 4-15: VICM RAM errors, records 15-16 on page 110 • Table 4-16: VSPA RAM errors, records 17-18 on page 111 • Table 4-17: VTGT_VSTR RAM errors, records 19-20 on page 112 • Table 4-18: VTGT_VRES RAM errors, records 21-22 on page 112 • Table 4-19: VTGT_SRCH RAM errors, records 23-24 on page 113 • Table 4-20: ITS RAM errors, records 25-26 on page 113 • 4.17.4.15 ITS command and translation error records 27+ on page 114 • Table 4-23: CC RAM errors, records 62-63 on page 130

The following table shows the Data field encoding for each error record and syndrome.

In the following table, the *Error record bank, by view* column is applicable only for GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1.

Table 5-106: GICT_ERR<n>MISC0.Data field encoding

Record	GICT_ERR<n>STATUS.IERR (syndrome)	GICT_ERR<n>STATUS.SERR	Value and description of GICT_ERR<n>MISC0.Data (other bits RES0) Always packed from 0 (lowest = 0)	Error record bank, by view
Software error (0)	0x0, SYN_ACE_BAD Illegal ACE5-Lite subordinate access.	0xE	AccessRnW, bit[12] AccessSparse, bit[11] AccessSize, bits[10:8] AccessLength, bits[7:0]	Access view
Software error (0)	0x1, SYN_PPI_PWRDWN Attempt to access a powered down Redistributor.	0xF	Redistributor, bits[24:16] Core, bits[8:0]	0
Software error (0)	0x2, SYN_PPI_PWRCHANGE Attempt to power down Redistributor rejected.	0xF	Redistributor, bits[24:16] Core, bits[8:0]	0
Software error (0)	0x4, SYN_PROPBASE_ACC Attempt to reprogram PROPBASE registers to a value that is not accepted because another value is already in use.	0xF	Core, bits[8:0]	Access view
Software error (0)	0x5, SYN_PENDBASE_ACC Attempt to reprogram PENDBASE registers to a value that is not accepted because another value is already in use.	0xF	Core, bits[8:0]	Access view
Software error (0)	0x7, SYN_WAKER_CHANGE Attempt to change GICR_WAKER abandoned due to handshake rules.	0xF	Core, bits[8:0]	Access view
Software error (0)	0x8, SYN_SLEEP_FAIL Attempt to put GIC to sleep failed because cores are not fully asleep.	0xF	Core, bits[8:0]	0
Software error (0)	0x9, SYN_PGE_ON_QUIESCE Core put to sleep before its Group enables were cleared.	0xF	Core, bits[8:0]	Access view
Software error (0)	0x10, SYN_SGI_NO_TGT SGI sent with no valid destinations.	0xE	Core, bits[8:0]	SGI source view
Software error (0)	0x11, SYN_SGI_CORRUPTED SGI corrupted without effect.	0x6	Core, bits[8:0]	0
Software error (0)	0x12, SYN_GICR_CORRUPTED Data was read from GICR register space that encountered an uncorrectable error.	0x6	GICT_ERR0ADDR is populated	View
Software error (0)	0x13, SYN_GICD_CORRUPTED Data was read from GICD register space that encountered an uncorrectable error.	0x6	GICT_ERR0ADDR is populated	0
Software error (0)	0x14, SYN_ITS_OFF Data was read from an ITS that is powered down.	0xF	GICT_ERR0ADDR is populated	Access view
Software error (0)	0x18, SYN_SPI_BLOCK. Attempt to access an SPI block that is not implemented.	0xE	Block, bits[4:0]	Access view

Record	GICT_ERR<n>STATUS.IERR (syndrome)	GICT_ERR<n>STATUS.SERR	Value and description of GICT_ERR<n>MISC0.Data (other bits RES0) Always packed from 0 (lowest = 0)	Error record bank, by view
Software error (0)	0x19, SYN_SPI_OOR Attempt to access a non-implemented SPI using (SET|CLR)SPI.	0xE	ID, bits[9:0]	Access view
Software error (0)	0x1A, SYN_SPI_NO_DEST_TGT An SPI has no legal target destinations.	0xF	ID, bits[9:0]	SPI view
Software error (0)	0x1B, SYN_SPI_NO_DEST_1OFN A 1 of N SPI cannot be delivered due to bad GICR_CTLR.DPG<0|1NS|1S> or GICR_CLASSR programming.	0xF	ID, bits[9:0]	SPI view
Software error (0)	0x1C, SYN_COL_OOR A collator message is received for a non-implemented SPI, or is larger than the number of owned SPIs in a multichip configuration.	0xF	ID, bits[9:0]	0
Software error (0)	0x1D, SYN_DEACT_IN A `Deactivate` command to a nonexistent SPI, or with incorrect groups set. `Deactivate` commands to LPI and nonexistent PPI are not reported.	0xE	None	Source PE view
Software error (0)	0x1E, SYN_SPI_CHIP_OFFLINE An attempt was made to send an SPI to an offline chip.	0xF	ID, bits[9:0]	SPI view
Software error (0)	0x25, SYN_VSGI_OFFLINE Pending vSGI to a vPEID mapped to an offline chip.	0xF	Chip [$\log_2$(chips)−1:0] ID (multi-hot) [15:0] vPEID[$\log_2$(vpes)−1:0]	1
Software error (0)	0x30, SYN_VSGI_UNMAPPED Pending vSGI to a vPEID that is not mapped.	0xF	ID (multi-hot) [15:0] vPEID[$\log_2$(vpes)−1:0]	1
Software error (0)	0x33, SYN_VSGI_LOST Pending vSGI to a vPEID that has inconsistent mapping information across multiple chips.	0xF	ID (multi-hot) [15:0] vPEID [$\log_2$(vpes)−1:0]	
Software error (0)	0x34, SYN_VPT_READ_FAIL An attempt was made to read the vPE configuration from the virtual Pending table, with an error received with the read response.	0x12	vPEID [$\log_2$(vpes)−1:0]	
Software error (0)	0x35, SYN_VPT_WRITE_FAIL An attempt was made to write the vPE configuration to the virtual Pending table, with an error received with the write response. The vICM reports bad write responses on the chip where the access occurs, rather than the chip that contains the ITS that generated the command or interrupt.	0x12	vPEID [$\log_2$(vpes)−1:0]	
Software error (0)	0x39, SYN_VPE_CFG_PTR_FAIL An attempt was made to access an indirect vPE Configuration table with an invalid level 2 pointer.	0xD	vPEID [$\log_2$(vpes)−1:0]	
Software error (0)	0x3A, SYN_VPE_CFG_TOP_READ_FAIL An attempt was made to read the level 1 of an indirect vPE Configuration table, with an error received with the read response.	0x12	vPEID [$\log_2$(vpes)−1:0]	

Record	GICT_ERR<n>STATUS.IERR (syndrome)	GICT_ERR<n>STATUS.SERR	Value and description of GICT_ERR<n>MISC0.Data (other bits RES0) Always packed from 0 (lowest = 0)	Error record bank, by view
Software error (0)	0x3B, SYN_VPE_CFG_LEAF_READ_FAIL An attempt was made to read the level 2 of an indirect vPE Configuration table or any vPE Configuration read when the table is not indirect, with an error received with the read response.	0x12	vPEID [log_2(vpes)−1:0]	
Software error (0)	0x3C, SYN_VPE_CFG_WRITE_FAIL An attempt was made to write the level 2 of an indirect vPE Configuration table or any vPE Configuration write when the table is not indirect, with an error received with the read response. The vICM reports bad write responses on the chip where the access occurs, rather than the chip that contains the ITS that generated the command or interrupt.	0x12	vPEID [log_2(vpes)−1:0]	
Software error (0)	0x3D, SYN_VPE_CFG_OVERFLOW A vPE Configuration table access was aborted due to table entry overflow in the address space.	0xD	vPEID [log_2(vpes)−1:0]	
Software error (0)	0x40, SYN_LPI_PROP_READ_FAIL An attempt was made to read properties for a single interrupt where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	1
Software error (0)	0x41, SYN_PT_PROP_READ_FAIL An attempt was made to read properties for a block of interrupts where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	1
Software error (0)	0x42, SYN_PT_COARSE_MAP_READ_FAIL An attempt was made to read the coarse map for a target where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16]	1
Software error (0)	0x43, SYN_PT_COARSE_MAP_WRITE_FAIL An attempt was made to write the coarse map for a target with an error received with the write response.	0x12	Virtual, bit[30] Target, bits[29:16]	1
Software error (0)	0x44, SYN_PT_TABLE_READ_FAIL An attempt was made to read a block of interrupts from a Pending table, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	1
Software error (0)	0x45, SYN_PT_TABLE_WRITE_FAIL An attempt was made to write-back a block of interrupts from a Pending table with an error received with the write response.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	1
Software error (0)	0x46, SYN_PT_SUB_TABLE_READ_FAIL An attempt was made to read a subblock of interrupts from a Pending table, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	1
Software error (0)	0x47, SYN_PT_TABLE_WRITE_FAIL_BYTE An attempt was made to write-back a subblock of interrupts from a Pending table, with an error received with the write response.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	1

Record	GICT_ERR<n>STATUS.IERR (syndrome)	GICT_ERR<n>STATUS.SERR	Value and description of GICT_ERR<n>MISC0.Data (other bits RES0) Always packed from 0 (lowest = 0)	Error record bank, by view
Software error (0)	0x48, SYN_DBL_PROP_READ_FAIL An attempt was made to read properties for a single doorbell, where an error response was received with the data.	0x12	Virtual, bit[30] Target, bits[29:16] ID, bits[15:0]	1
Software error (0)	0x50, SYN_VPROPBASER_DATA An attempt was made to program additional GICR_VPROPBASER.Valid bits with data mismatching GICR_VCFGBASER.	0xF	CPU [$\log_2$(cpus)−1:0]	1
Software error (0)	0x52, SYN_VERRR_BUSY An attempt was made to access GICR_VERRR while the register is busy from a previous operation.	0xF	CPU [$\log_2$(cpus)−1:0]	1
Software error (0)	0x53, SYN_VERRR_ALLOC An attempt was made to access GICR_VERRR while there is no vPE Configuration table allocation.	0xF	CPU [$\log_2$(cpus)−1:0]	1
Software error (0)	0x54, SYN_VERRR_VPE_OOR A request was made to GICR_VERRR with a vPEID that is out of range.	0xE	CPU [$\log_2$(cpus)−1:0]	1
Software error (0)	0x56, SYN_VSGIR_ALLOC An attempt was made to access GICR_VSGIR while there is no vPE Configuration table allocation.	0xF	CPU [$\log_2$(cpus)−1:0]	1
Software error (0)	0x57, SYN_VSGIR_VPE_OOR A request was made to GICR_VSGIR with a vPEID that is out of range.	0xE	CPU [$\log_2$(cpus)−1:0]	1
Software error (0)	0x58, SYN_VINV_BUSY An attempt was made to access GICR_VINVCHIPR while the register is busy from a previous operation.	0xF	CPU [$\log_2$(cpus)−1:0]	1
Software error (0)	0x59, SYN_VINV_ALLOC An attempt was made to access GICR_VINVCHIPR while there is no vPE Configuration table allocation.	0xF	CPU [$\log_2$(cpus)−1:0]	1
Software error (0)	0x60, SYN_ACE_CC_BAD Illegal cross-chip ACE5-Lite subordinate access.	0xE	Access chip, [15:4] Access opcode, [3:0]	0
Software error (0)	0x70, SYN_ITS_REG_INV_BUSY An attempt was made to invalidate an interrupt while register busy.	0xF	CPU, [$\log_2$(cores) − 1:0] Data, bits[15:0]	Access view
Software error (0)	0x71, SYN_ITS_REG_INV_OOR An attempt was made to invalidate an OOR interrupt.	0xE	CPU, [$\log_2$(cores) − 1:0] Data, bits[15:0]	Access view
Correctable SPI RAM errors (1) Uncorrectable SPI RAM errors (2)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-8: SPI RAM errors, records 1-2 on page 106	0
Correctable SGI RAM errors (3)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-9: SGI RAM errors, records 3-4 on page 107	0

Record	GICT_ERR<n>STATUS.IERR (syndrome)	GICT_ERR<n>STATUS.SERR	Value and description of GICT_ERR<n>MISC0.Data (other bits RES0) Always packed from 0 (lowest = 0)	Error record bank, by view
Uncorrectable SGI RAM errors (4)				
Correctable TGT_SPI cache errors (5)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-10: TGT_SPI RAM errors, records 5-6 on page 107	0
Uncorrectable TGT_SPI cache errors (6)				
Correctable PPI RAM errors (7)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-11: PPI RAM errors, records 7-8 on page 108	0
Uncorrectable PPI RAM errors (8)				
Correctable LPI RAM errors (9)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-12: LPI RAM errors, records 9-10 on page 109	0
Uncorrectable LPI RAM errors (10)				
Correctable PTS RAM error (11)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-13: PTS RAM errors, records 11-12 on page 109	0
Uncorrectable PTS RAM error (12)				
Correctable TGT_LPI RAM error (13)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-14: TGT_LPI RAM errors, records 13-14 on page 110	0
Uncorrectable TGT_LPI RAM error (14)				
Correctable VICM RAM error (15)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-15: VICM RAM errors, records 15-16 on page 110	0
Uncorrectable VICM RAM error (16)				
Correctable VSPA RAM error (17)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-16: VSPA RAM errors, records 17-18 on page 111	0

Record	GICT_ERR<n>STATUS.IERR (syndrome)	GICT_ERR<n>STATUS.SERR	Value and description of GICT_ERR<n>MISC0.Data (other bits RES0) Always packed from 0 (lowest = 0)	Error record bank, by view
Uncorrectable VSPA RAM error (18)				
Correctable VTGT_VSTR RAM error (19)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-17: VTGT_VSTR RAM errors, records 19-20 on page 112	0
Uncorrectable VTGT_VSTR RAM error (20)				
Correctable VTGT_VRES RAM error (21)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-18: VTGT_VRES RAM errors, records 21-22 on page 112	0
Uncorrectable VTGT_VRES RAM error (22)				
Correctable VTGT_SRCH RAM error (23)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-19: VTGT_SRCH RAM errors, records 23-24 on page 113	0
Uncorrectable VTGT_SRCH RAM error (24)				0
Correctable error from ITS RAM (25)	**0x00** A real error **0x01** An injected error	0x6	See Table 4-20: ITS RAM errors, records 25-26 on page 113	0
Uncorrectable error from ITS RAM (26)				
Command or translation error in ITS (27+)	**0x00** Architectural **0x01** Non-architectural	0x1	ITS 24-bit syndrome. See 4.17.4.15 ITS command and translation error records 27+ on page 114.	0
Correctable error from CC RAM (62)	**0x00** A real error **0x01** An injected error	0x7	See Table 4-23: CC RAM errors, records 62-63 on page 130	0
Uncorrectable error from CC RAM (63)				

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_ERR<n>MISC0 is accessible only by Secure accesses.

5.10.6 GICT_ERR<n>MISC1, Error Record Miscellaneous Register 1

This register contains the data value of an uncorrectable error in the LPI RAM, TGT_LPI RAM, or ITS software information. The register is not present for other error records.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

If GICT_ERR<n>STATUS.MV == 1, then GICT_ERR<n>MISC1 ignores writes.

Bit descriptions

Figure 5-78: GICT_ERR<n>MISC1 bit assignments

Table 5-107: GICT_ERR<n>MISC1 bit descriptions

Bits	Name	Description
[63:x+1]	-	Reserved, RAZ
[x:0]	INFO	Contains the corrupted data that is read from the RAM. The value x depends on the width of the RAM, which is set during the configuration of GIC-720AE.

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_ERR<n>MISC1 is accessible only by Secure accesses.

5.10.7 GICT_ERRGSR, Error Group Status Register

This register shows the status of the GIC-720AE Armv8.2 RAS architecture-compliant error records for correctable and uncorrectable RAM ECC errors, ITS command and translation errors, and uncorrectable software errors.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-79: GICT_ERRGSR bit assignments

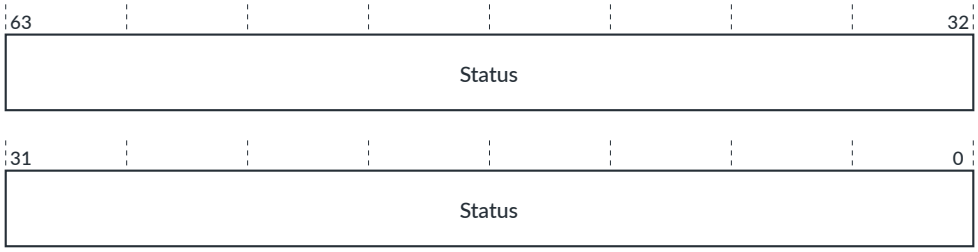


Table 5-108: GICT_ERRGSR bit descriptions

Bits	Name	Description
[n]	Status	Indicates the status of error record n, where n is 0-27+ depending on the configuration: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_ERRGSR is accessible only by Secure accesses.

5.10.8 GICT_IIDR, Trace Implementer Identification Register

This register provides information about the implementer and revision of the trace page.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-80: GICT_IIDR bit assignments

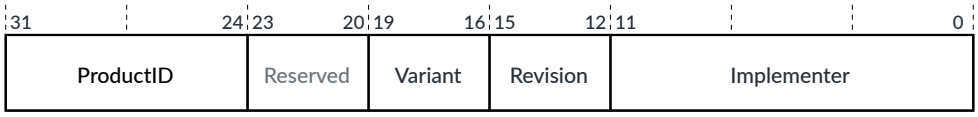


Table 5-109: GICT_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **0x07** GIC-720AE
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: **0x0** r0 **0x1** r1 **0x2** r2
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: **0x0** p0 **0x1** p1
[11:0]	Implementer	Identifies the implementer: **0x43B** Arm

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_IIDR is accessible only by Secure accesses.

5.10.9 GICT_ERRIRQCR<n>, Error Interrupt Configuration Registers

GICT_ERRIRQCR0 controls which SPI is generated when a fault handling interrupt occurs.
GICT_ERRIRQCR1 controls which SPI is generated when an error recovery interrupt occurs.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-81: GICT_ERRIRQCR<n> bit assignments

Table 5-110: GICT_ERRIRQCR<n> bit descriptions

Bits	Name	Description
[31:11]	-	Reserved, RAZ
[10:0]	SPIID	Sets the SPI ID that the GIC generates when a fault handling interrupt occurs (<n>==0) or when an error recovery interrupt occurs (<n>==1). If the value is less than 32, out of range, or not owned on chip for multichip configurations, the register updates to 0 and no internal delivery occurs. Set this field to 0 when the interrupt routes externally to a core that does not receive interrupts directly from the GIC such as a central system control processor. **Note:** The behavior is unpredictable if software attempts to share the same interrupt ID in GICT_ERRIRQCRn with an external source using either: • An SPI wire. • The GICD_SETSPI_NSR or GICD_SETSPI_SR registers. In a multichip configuration, only program the SPIID field to an SPI ID that the chip owns. The relevant GICD_CHIPRn register controls the SPI ownership. We recommend that if these registers are used, then the SPI must not be used for another device, either with a wire or as a message-based interrupt.

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_ERRIRQCR<n> is accessible only by Secure accesses.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, GICT_ERRIRQCR<n> is accessible for view 0 only.

5.10.10 GICT_DEVID, Device Configuration register

This register returns information about the configuration of the GIC-720AE GICT such as whether an LPI or ITS is available.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-82: GICT_DEVID bit assignments

Table 5-111: GICT_DEVID bit descriptions

Bits	Name	Description
[31:16]	-	Reserved, RAZ
[15:0]	NUM	Returns the index of the last error record, plus one: **9** No LPI available. **28-60** LPI available with one or more ITS. The number of ITSs = NUM − 28. **64** This value occurs when the GIC has an ACE5-Lite cross-chip interface.

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_DEVID is accessible only by Secure accesses.

5.10.11 GICT_PIDR2, Peripheral ID2 Register

This register returns byte[2] of the peripheral ID. The GICT_PIDR2 register is part of the set of trace and debug peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.10 GICT register summary on page 249 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-83: GICT_PIDR2 bit assignments

Table 5-112: GICT_PIDR2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:4]	ArchRev	Identifies the version of the GIC architecture with which the trace and debug block complies: **0x3** GICv3 **0x4** GICv4
[3]	JEDEC	Indicates that a JEDEC-assigned JEP106 identity code is used.
[2:0]	DES_1	Bits[6:4] of the JEP106 identity code. Bits[3:0] of the JEP106 identity code are assigned to GICT_PIDR1[7:4].

Accessibility

If GICD_SAC.GICTNS == 0, then GICT_PIDR2 is accessible only by Secure accesses.

5.11 GICP register summary

The GIC-720AE Performance Monitoring Unit functions are controlled through registers that are identified with the prefix GICP.

The GICD_SAC.GICPNS bit controls whether Non-secure software can access the GICP registers.

Table 5-113: GICP register summary

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x000 + (n × 4)	GICP_EVCNTRn	RW	Unknown	32	Event Counter Registers, n = 0-4	No

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x400 + (n × 4)	GICP_EVTYPERn	RW	Unknown	32	Event Type Configuration Registers, n = 0-4	No
0x600 + (n × 4)	GICP_SVRn	RO	Unknown	32	Shadow Value Registers, n = 0-4	No
0xA00 + (n × 4)	GICP_FRn	RW	Unknown	32	Filter Registers, n = 0-4	No
0xC00	GICP_CNTENSET0	RW	0x0	64	Counter Enable Set Register	No
0xC20	GICP_CNTENCLR0	RW	0x0	64	Counter Enable Clear Register	No
0xC40	GICP_INTENSET0	RW	0x0	64	Interrupt Contribution Enable Set Register 0	No
0xC60	GICP_INTENCLR0	RW	0x0	64	Interrupt Contribution Enable Clear Register 0	No
0xC80	GICP_OVSCLR0	RW	0x0	64	Overflow Status Clear Register 0	No
0xCC0	GICP_OVSSET0	RW	0x0	64	Overflow Status Set Register 0	No
0xD88	GICP_CAPR	WO	-	32	Counter Shadow Value Capture Register	No
0xE00	GICP_CFGR	RO	0x401F04	32	Configuration Information Register	No
0xE04	GICP_CR	RW	0x0	32	Control Register	No
0xE08	GICP_IIDR	RO	0x070nn43B The nn value depends on the r*x*p*y* identifier.	32	PMU Implementer Identification Register	No
0xE50	GICP_IRQCR	RW	0x0	32	Interrupt Configuration Register	No
0xFB8	GICP_PMAUTHSTATUS	RO	0x088	32	Authentication Status register	No
0xFBC	GICP_PMDEVARCH	RO	0x47702A56	32	Device Architecture register	No
0xFCC	GICP_PMDEVTYPE	RO	0x56	32	Device Type register	No
0xFD0	GICP_PIDR4	RO	0x44	32	Peripheral ID 4 Register	No
0xFD4	GICP_PIDR5	RO	0x00	32	Peripheral ID 5 Register	No
0xFD8	GICP_PIDR6	RO	0x00	32	Peripheral ID 6 Register	No
0xFDC	GICP_PIDR7	RO	0x00	32	Peripheral ID 7 Register	No
0xFE0	GICP_PIDR0	RO	0x96	32	Peripheral ID 0 Register	No
0xFE4	GICP_PIDR1	RO	0xB4	32	Peripheral ID 1 Register	No
0xFE8	GICP_PIDR2	RO	0x3B	32	Peripheral ID 2 Register	No
0xFEC	GICP_PIDR3	RO	0x00	32	Peripheral ID 3 Register	No
0xFF0	GICP_CIDR0	RO	0x0D	32	Component ID 0 Register	No
0xFF4	GICP_CIDR1	RO	0xF0	32	Component ID 1 Register	No
0xFF8	GICP_CIDR2	RO	0x05	32	Component ID 2 Register	No
0xFFC	GICP_CIDR3	RO	0xB1	32	Component ID 3 Register	No

5.11.1 GICP_EVCNTRn, Event Counter Registers

These registers contain the values of event counter `n`. The GIC-720AE supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-84: GICP_EVCNTRn bit assignments

Table 5-114: GICP_EVCNTRn bit descriptions

Bits	Name	Description
[31:0]	COUNT	Counter value. If the counter is enabled, the counter value increments when an event matching GICP_EVTYPERn.EVENT occurs.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_EVCNTRn is accessible only by Secure accesses.

5.11.2 GICP_EVTYPERn, Event Type Configuration Registers

These registers configure which events that event counter `n` counts. The GIC-720AE supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit

Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-85: GICP_EVTYPERn bit assignments

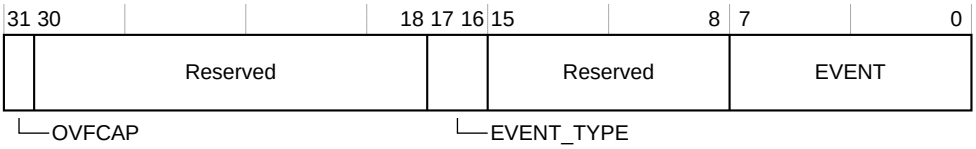


Table 5-115: GICP_EVTYPERn bit descriptions

Bits	Name	Description
[31]	OVFCAP	When set to 1, an overflow of counter `n` triggers a capture if GICP_CAPR.CAPTURE is set.
[30:18]	-	Reserved
[17:16]	EVENT_TYPE	Event tracking type: **0b00** Count events **0b10** MaximumEvent **0b01, 0b11** Reserved
[15:8]	-	Reserved
[7:0]	EVENT	Event identifier. See Table 5-116: GICP_EVTYPERn.EVENT field encoding on page 272. All events reset to an unknown value. Registers corresponding to unimplemented counters are RES0.

The following table shows the events that the GIC can count. The mask column indicates whether Secure events can be masked when GICD_SAC.SPF = 1 and GICD_CTLR.DS == 0.

In the following table, the View filter column is relevant only when the GIC configuration supports multi view, that is, when GICD_CFGID.VIEW == 1. The view filter is set by GICD_SAC.GICPVIEW. Software can observe the events only when the view filter matches GICD_SAC.GICPVIEW or when GICD_SAC.GICPVIEW == 0.

Table 5-116: GICP_EVTYPERn.EVENT field encoding

EventID	Event	Description	Mask	View filter	Filter
0x0	CLK	Clock cycle	Unmasked	-	None
0x1	CLK_NG	Clock cycle that prevents Q-Channel clock gating.	Unmasked	-	None
0x2-0x3	-	Reserved	-	-	-
0x4	DN_MSG_PHY	Downstream message to core excluding PPIs.	Masked	0	TargetVP

EventID	Event	Description	Mask	View filter	Filter
0x5	DN_SET_PHY	Set to core SPIs, LPIs, and doorbells.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view, LPI view 1	TargetVP/ID range
0x6	DN_SET1OFN_PHY	Set to core, which is a 1 of N interrupt.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	TargetVP/ID range
0x7	-	Reserved	-	-	-
0x8	UP_MSG_PHY	Upstream message from core.	Masked	0	TargetVP
0x9	UP_ACT_SPI	Upstream activate.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	TargetVP/ID range
0xA	UP_REL_PHY	Upstream release.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	Target
0xB	UP_ACT_LPI	Upstream activate of LPI.	Unmasked	1	TargetVP/ID range
0xC	UP_SET_COMP_PHY	A set followed by an activate. This event counts the set and then decrements on release.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view, LPI view 1	Target
0xD	UP_DEACT	Upstream deactivate. SPIs only.	The event is masked when the Deactivate packet has either Group 0 or Secure Group 1 set.	RD view	TargetVP/ID range
0xE	UP_ACT_DBL	Upstream activate of doorbell.	Unmasked	1	TargetVP(vPE)/ID range
0x10	SGI_BRD	Broadcast SGI messages. Target = source.	The event is masked when the Generate SGI packet has the NS bit set to 0.	RD view	TargetVP/ID range
0x11	SGI_TAR	Targeted SGI messages. Target = source.	The event is masked when the Generate SGI packet has the NS bit set to 0.	RD view	TargetVP/ID range
0x12	SGI_ALL	All SGI messages. Target = source.	The event is masked when the Generate SGI packet has the NS bit set to 0.	RD view	TargetVP/ID range
0x13	SGI_ACC	Accepted SGI. Target = source.	The event is masked when the Generate SGI packet has the NS bit set to 0.	RD view	TargetVP/ID range

EventID	Event	Description	Mask	View filter	Filter
0x14	SGI_BRD_CC_IN	Broadcast SGI message from cross-chip.	The event is masked when the Generate SGI packet has the NS bit set to 0.	0	ID range/Chip
0x15	SGI_TAR_CC_IN	Targeted SGI message from cross-chip.	The event is masked when the Generate SGI packet has the NS bit set to 0.	0	ID range/Chip
0x16	SGI_TAR_CC_OUT	Targeted SGI sent cross-chip.	The event is masked when the Generate SGI packet has the NS bit set to 0.	0	Chip/ID range
0x17	SGI_CC_OUT	Any SGI being sent cross-chip.	The event is masked when the Generate SGI packet has the NS bit set to 0.	0	Chip
0x18	SGI_CC_OUT_RESP	Response from any outgoing SGI.	The event is masked when the Generate SGI packet has the NS bit set to 0.	0	Chip
0x20	ITS_NLL_LPI	Incoming LPI	Unmasked	1	TargetVP/ID range/ITS
0x21	ITS_LL_LPI	Incoming low latency LPI.	Unmasked	1	TargetVP/ID range/ITS
0x22	ITS_LPI	Incoming LPI (or low latency).	Unmasked	1	TargetVP/ID range/ITS
0x23	ITS_LPI_CMD	Incoming LPI command	Unmasked	1	TargetVP/ID range/ITS
0x24	ITS_DID_MISS	Number of DeviceID cache misses.	Unmasked	1	TargetVP/ID range/ITS
0x25	ITS_VID_MISS	Number of EventID cache misses.	Unmasked	1	TargetVP/ID range/ITS
0x26	ITS_COL_MISS	Number of Collection cache misses.	Unmasked	1	TargetVP/ID range/ITS
0x27	ITS_LAT	Latency of the ITS transaction.	Unmasked	1	TargetVP/ID range/ITS
0x28	ITS_MPFA	Number of free slots during translation	Unmasked	1	TargetVP/ID range/ITS
0x29	LPI_CC_OUT	LPI sent cross-chip.	Unmasked	0	ID range/Chip
0x2A	LPI_CMD_CC_OUT	LPI command sent cross-chip.	Unmasked	0	ID range/Chip
0x2B	LPI_CC_IN	LPI coming in from cross-chip.	Unmasked	0	ID range/Chip
0x2C	LPI_CMD_CC_IN	LPI command coming in from cross-chip.	Unmasked	0	ID range/Chip
0x2D	LPI_CC_OUT_RESP	Response to LPI sent cross-chip.	Unmasked	0	Chip
0x2E	LPI_CMD_CC_OUT_RESP	Response to LPI command sent cross-chip.	Unmasked	0	Chip
0x30	LPI_OWN_STORED	LPI stored in own location. Prevents clock gating and Q-Channel clock gating.	Unmasked	1	-
0x31	LPI_OOL_STORED	LPI stored out of location. Prevents clock gating and Q-Channel clock gating.	Unmasked	1	-
0x32	LPI_HIT_EN	LPI property read cache hit enabled. Uses the filter from counter 0 only.	Unmasked	1	TargetVP/ID range
0x33	LPI_HIT_DIS	LPI property read cache hit disabled. Uses the filter from counter 0 only.	Unmasked	1	TargetVP/ID range
0x34	LPI_HIT	LPI property read cache hit. Uses the filter from counter 0 only.	Unmasked	1	TargetVP/ID range

EventID	Event	Description	Mask	View filter	Filter
0x35	LPI_MATCH	LPI coalesced. Uses the filter from counter 0 only.	Unmasked	1	TargetVP/ID range
0x36	LPI_FAS	Number of slots free on new LPI.	Unmasked	1	None
0x37	LPI_PROP_EN	Enabled LPI property fetch. Uses the filter from counter 0.	Unmasked	1	TargetVP/ID range
0x38	LPI_PROP_DIS	Disabled LPI property fetch. Uses the filter from counter 0.	Unmasked	1	TargetVP/ID range
0x39	LPI_PROP	LPI property fetch. Uses the filter from counter 0.	Unmasked	1	TargetVP/ID range
0x50	SPI_COL_MSG	New message from SPI Collator.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	ID range
0x51	SPI_ENABLED	SPI enabled (new SPI or register access if pending).	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	ID range
0x52	SPI_DISABLED	SPI disabled (new SPI that is disabled or register access if pending).	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	ID range
0x53	SPI_PENDING_SET	New SPI pending valid.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	ID range
0x54	SPI_PENDING_CLR	SPI pending bit cleared.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	ID range
0x55	SPI_MATCH	Collated edge-based SPI. Excludes collation in the SPI Collator.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	SPI view	ID range
0x57	SPI_CC_IN	SPI from remote chip.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	0	ID range/Chip
0x58	SPI_CC_OUT	SPI sent to remote chip.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	0	ID range/Chip

EventID	Event	Description	Mask	View filter	Filter
0x59	SPI_CC_OUT_RESP	Response to SPI sent to remote chip.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	0	Chip
0x5A	SPI_CC_DEACT	SPI deactivate message sent.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	0	ID range/Chip
0x5B	SPI_CC_DEACT_RESP	Response to deactivate sent cross-chip.	The event is masked when it corresponds to an interrupt that is either Group 0 or Secure Group 1.	0	Chip
0x60	PT_IN_EN	Enabled interrupt written to Pending table.	Unmasked	1	TargetVP/ID range
0x61	PT_IN_DIS	Disabled interrupt written to Pending table.	Unmasked	1	TargetVP/ID range
0x62	PT_PRI	Priority of interrupt written to Pending table.	Unmasked	1	TargetVP/ID range
0x63	PT_IN	Interrupt written to Pending table.	Unmasked	1	TargetVP/ID range
0x64	PT_MATCH	Interrupt already set in Pending table.	Unmasked	1	TargetVP/ID range
0x65	PT_OUT_EN	Enabled interrupt taken out of Pending table (also covered PT_MATCH when enabled).	Unmasked	1	TargetVP/ID range
0x66	PT_OUT_DIS	Disabled interrupt taken out of Pending table (also covered PT_MATCH when disabled).	Unmasked	1	TargetVP/ID range
0x67	PT_OUT	Interrupt taken out of Pending table (also covered PT_MATCH).	Unmasked	1	TargetVP/ID range
0x70	VSGI_CC	vSGI sent cross-chip.	Unmasked	0	TargetVP/Chip
0x71	VSGI_CC_RESP	vSGI cross-chip response.	Unmasked	0	Chip
0x72	VSGI_IN_RAM	vSGI stored in RAM.	Unmasked	1	TargetVP
0x73	VLPI_BUFF_FILL	Number of buffers used on vLPI arriving.	Unmasked	1	-
0x74	VLPI_REJECT	vLPI sent cross-chip being rejected.	Unmasked	1	TargetVP/Chip
0x75	VSGI_REJECT	vSGI sent cross-chip being rejected.	Unmasked	1	TargetVP/Chip
0x76	VCMD_REJECT	Virtual command sent cross-chip being rejected.	Unmasked	1	TargetVP/Chip
0x78	RES_START	Residency change start.	Unmasked	1	TargetVP
0x79	RES_COMP	Residency change end.	Unmasked	1	TargetVP
0x80	ACC	Counter(n – 1) – counter(n – 2) every cycle. Prevents clock gating and Q-Channel clock gating.	Unmasked	-	None

EventID	Event	Description	Mask	View filter	Filter
0x81	OFLOW	Overflow of counter n − 1. Overflow counters cannot count overflows of the counters that are using the OFLOW event.	Unmasked	-	None
0x88	DN_SET_VIRT	Virtual set command.	Unmasked	1	TargetVP(PHY)/ID range
0x89	UP_REL_VIRT	Virtual release	Unmasked	1	TargetVP(PHY)
0x8A	UP_ACT_VLPI	Activate of vLPI.	Unmasked	1	TargetVP(PHY)/ID range
0x8B	UP_ACT_VSGI	Activate of vSGI.	Unmasked	1	TargetVP(PHY)/ID range
0x8C	UP_SET_COMP_VIRT	A set followed by an activate. This event counts the set and then decrements on release.	Unmasked	1	Target(PHY)
0x90	RLT_SPI_SET	Set to real-time SPIs. This event always uses the filter in GICP_FR3, irrespective of the counter that is used. For example, if using counter 4, then the GIC uses GICP_FR3 and ignores GICP_FR4.	Masked by group.	SPI view	Target/ID. To filter this event, you must precisely match the target or ID, otherwise the behavior is unpredictable.
0x91	RLT_SPI_ACT	Upstream activate for real-time SPIs. This event always uses the filter in GICP_FR3, irrespective of the counter that is used.	Masked by group.	SPI view	Target/ID. To filter this event, you must precisely match the target or ID, otherwise the behavior is unpredictable.
0x92	RLT_SPI_REL	Upstream release for real-time SPIs. This event always uses the filter in GICP_FR3, irrespective of the counter that is used.	Masked by group.	SPI view	Target/ID. To filter this event, you must precisely match the target or ID, otherwise the behavior is unpredictable.
0x93	RLT_SPI_DEACT	Upstream deactivate for real-time SPIs. This event always uses the filter in GICP_FR3, irrespective of the counter that is used.	Not reported if filter is set.	RD view	Target/ID. To filter this event, you must precisely match the target or ID, otherwise the behavior is unpredictable.
0x94	RLT_SPI_PEND_CNT	Pending real-time SPIs count at every cycle. Selecting this event prevents the Q-Channel from accepting low-power requests. This event always uses the filter in GICP_FR3, irrespective of the counter that is used.	Masked by group.	SPI view	Target/ID range
0x95	RLT_SPI_ACT_CNT	Activate count for real-time SPIs at every cycle. Selecting this event prevents the Q-Channel from accepting low-power requests. This event always uses the filter in GICP_FR3, irrespective of the counter that is used.	Masked by group.	SPI view	Target/ID range

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_EVTYPERn is accessible only by Secure accesses.

5.11.3 GICP_SVRn, Shadow Value Registers

These registers contain the shadow value of event counter `n`. The GIC-720AE supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-86: GICP_SVRn bit assignments

Table 5-117: GICP_SVRn bit descriptions

Bits	Name	Description
[31:0]	COUNT	Captured counter value. This field holds the captured counter values of the corresponding entry in GICP_EVCNTRn.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_SVRn is accessible only by Secure accesses.

5.11.4 GICP_FRn, Filter Registers

These registers configure the filtering of event counter `n`. The GIC-720AE supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-87: GICP_FRn bit assignments

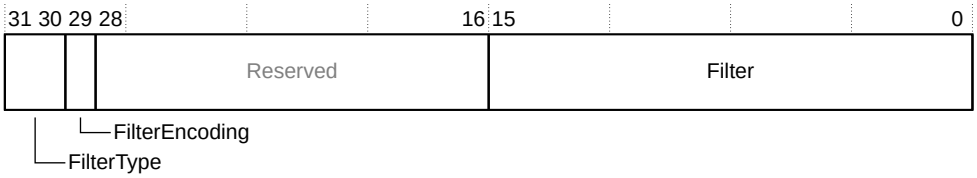


Table 5-118: GICP_FRn bit descriptions

Bits	Name	Description
[31:30]	FilterType	Filter type: **0b00** Filter on core or vPE or both. **0b01** Filter on INTID. **0b10** Filter on chip or ITS. **0b11** Reserved, no effect.
[29]	FilterEncoding	**0** Filter on range. **1** Filter on an exact match.
[28:16]	-	Reserved

Bits	Name	Description
[15:0]	Filter	If the corresponding GICP_EVTYPERn.EVENT indicates an event that cannot be filtered, then the value in this register is ignored. When FilterEncoding == 1, counter `n` counts events that are associated only with an exact match of the FilterType. When FilterEncoding == 0, this field is encoded so that the first LSB that is zero, indicates the uppermost of a contiguous span of least significant FilterType content bits, that the GIC ignores for the purposes of matching. For example, setting Filter to: • `0b11110111_11110111` matches with values of `0b11110111_1111xxxx` for FilterType content. • `0b11110111_11110110` matches with values of `0b11110111_1111011x` for FilterType content. • `0b11110101_11111111` matches with values of `0b111101xx_xxxxxxxx` for FilterType content. For events with filtering that is specified as TargetVP in Table 5-116: GICP_EVTYPERn.EVENT field encoding on page 272, then the top 2 bits of the filter value have alternative functionality: **Filter bit[15]** 0 = Use vPE in match. 1 = Do not use vPE. Virtual events fail in the filter. **Filter bit[14]** 0 = Use PE in match. 1 = Do not use PE. Physical events fail in the filter.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_FRn is accessible only by Secure accesses.

5.11.5 GICP_CNTENSET0, Counter Enable Set Register 0

These registers contain the counter enables for each event counter. The GIC-720AE supports five event counters.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-88: GICP_CNTENSET0 bit assignments

Table 5-119: GICP_CNTENSET0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ
[4:0]	CNTEN	Counter enable. The CNTEN[n] bit is the enable for counter n. This field resets to an unknown value. Reads return the state of the counter enables. Writing: **Bit[n] = 1** Sets the enable for counter n. **Bit[n] = 0** No effect. To disable a counter, use GICP_CNTENCLR0. Counter n is enabled when CNTEN[n] == 1 and GICP_CR.E == 1.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_CNTENSET0 is accessible only by Secure accesses.

5.11.6 GICP_CNTENCLR0, Counter Enable Clear Register 0

This register contains the counter disables for each event counter. The GIC-720AE supports five event counters.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-89: GICP_CNTENCLR0 bit assignments

Table 5-120: GICP_CNTENCLR0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ
[4:0]	CNTEN	Counter disable. The CNTEN[n] bit is the disable for counter n. This field resets to an unknown value. Reads return the state of the counter enables. Writing: **Bit[n] = 1** Disables counter n. **Bit[n] = 0** No effect. To enable a counter, use GICP_CNTENSET0. Counter n is disabled when CNTEN[n] == 0 or GICP_CR.E == 0.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_CNTENCLR0 is accessible only by Secure accesses.

5.11.7 GICP_INTENSET0, Interrupt Contribution Enable Set Register 0

This register contains the set mechanism for the counter interrupt contribution enables. The GIC-720AE supports five counters, n = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-90: GICP_INTENSET0 bit assignments

Table 5-121: GICP_INTENSET0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ

Bits	Name	Description
[4:0]	INTEN	Interrupt enable. The INTEN[n] bit is the interrupt enable for counter n. This field resets to an unknown value. Reads return the state of the interrupt enables. Writing: **Bit[n] = 1** Sets the interrupt enable for counter n. **Bit[n] = 0** No effect. To disable a counter interrupt enable, use GICP_INTENCLR0. The interrupt enable for counter n is enabled when INTEN[n] == 1 and GICP_CR.E == 1. Overflow of counter n sets GICP_OVSSET0.OVS[n] to 1 and that triggers the PMU interrupt if INTEN[n] == 1.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_INTENSET0 is accessible only by Secure accesses.

5.11.8 GICP_INTENCLR0, Interrupt Contribution Enable Clear Register 0

This register contains the clear mechanism for the counter interrupt contribution enables. The GIC-720AE supports five counters, n = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-91: GICP_INTENCLR0 bit assignments

Table 5-122: GICP_INTENCLR0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ

Bits	Name	Description
[4:0]	INTEN	Interrupt enable. The INTEN[n] bit is the interrupt disable for counter n. This field resets to an unknown value. Reads return the state of the interrupt enables. Writing: **Bit[n] = 1** Clears the interrupt enable for counter n. **Bit[n] = 0** No effect. To set a counter interrupt enable, use GICP_INTENSET0.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_INTENCLR0 is accessible only by Secure accesses.

5.11.9 GICP_OVSCLR0, Overflow Status Clear Register 0

This register provides the clear mechanism for the counter overflow status bits and provides read access to the counter overflow status bit values. The GIC-720AE supports five counters, n = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-92: GICP_OVSCLR0 bit assignments

Table 5-123: GICP_OVSCLR0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ

Bits	Name	Description
[4:0]	OVS	Overflow status. The OVS[n] bit is the overflow clear for counter n. This field resets to zero. Reads return the state of the overflow status bits. Writing: **Bit[n] = 1** Clears the overflow status for counter n. **Bit[n] = 0** No effect. To set a counter overflow status, use GICP_OVSSET0. Overflow of counter n, that is a transition past the maximum unsigned value of the counter that causes the value to wrap and become zero, sets the corresponding OVS bit. In addition, this event can trigger the PMU interrupt and cause a capture of the PMU counter values, see 5.11.2 GICP_EVTYPERn, Event Type Configuration Registers on page 271.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_OVSCLR0 is accessible only by Secure accesses.

5.11.10 GICP_OVSSET0, Overflow Status Set Register 0

This register provides the set mechanism for the counter overflow status bits and provides read access to the counter overflow status bit values. The GIC-720AE supports five counters, n = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-93: GICP_OVSSET0 bit assignments

Table 5-124: GICP_OVSSET0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ

Bits	Name	Description
[4:0]	OVS	Overflow status. The OVS[n] bit is the overflow set for counter n. This field resets to zero. Reads return the state of the overflow status bits. Writing: **Bit[n] = 1** Sets the overflow status for counter n. **Bit[n] = 0** No effect. To clear a counter overflow status, use GICP_OVSCLR0. When the agent controlling the GIC-720AE sets an OVS bit, it is similar to an OVS bit being set because of a counter overflow. Setting the OVS bit triggers the overflow interrupt if it is enabled.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_OVSSET0 is accessible only by Secure accesses.

5.11.11 GICP_CAPR, Counter Shadow Value Capture Register

This register controls the counter shadow value capture mechanism.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-94: GICP_CAPR bit assignments

Table 5-125: GICP_CAPR bit descriptions

Bits	Name	Description	Type
[31:1]	-	Reserved	-
[0]	CAPTURE	A write of 1 triggers a capture of all values within the PMU into their respective shadow registers. A write of 0 has no effect. See Snapshot on page 97 for information about other snapshot event triggers.	WO

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_CAPR is accessible only by Secure accesses.

5.11.12 GICP_CFGR, Configuration Information Register

This register returns information about the PMU implementation.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-95: GICP_CFGR bit assignments

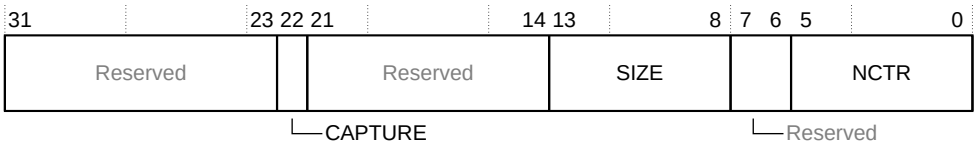


Table 5-126: GICP_CFGR bit descriptions

Bits	Name	Description
[31:23]	-	Reserved, RAZ
[22]	CAPTURE	Returns 1, to indicate that the GIC supports capture.
[21:14]	-	Reserved, RAZ
[13:8]	SIZE	Returns 31, to indicate that the GIC supports 32-bit counters.
[7:6]	-	Reserved, RAZ
[5:0]	NCTR	Returns 4, to indicate that the GIC provides five counters.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_CFGR is accessible only by Secure accesses.

5.11.13 GICP_CR, Control Register

This register controls whether all counters are enabled or disabled.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-96: GICP_CR bit assignments

Table 5-127: GICP_CR bit descriptions

Bits	Name	Description
[31:1]	-	Reserved
[0]	E	Global counter enable: **0** No events are counted and the values in GICP_EVCNTRn do not change. **1** The counters are enabled. Resets to 0. This bit takes precedence over the GICP_CNTENSET0.CNTEN bits.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_CR is accessible only by Secure accesses.

5.11.14 GICP_IIDR, PMU Implementer Identification Register

This register provides information about the implementer and revision of the PMU page.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-97: GICP_IIDR bit assignments

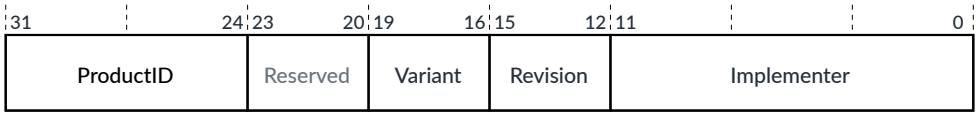


Table 5-128: GICP_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **0x07** GIC-720AE
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: **0x0** r0 **0x1** r1 **0x2** r2
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: **0x0** p0 **0x1** p1
[11:0]	Implementer	Identifies the implementer: **0x43B** Arm

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_IIDR is accessible only by Secure accesses.

5.11.15 GICP_IRQCR, Interrupt Configuration Register

This register controls which SPI is generated when a PMU overflow interrupt occurs.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-98: GICP_IRQCR bit assignments

Table 5-129: GICP_IRQCR bit descriptions

Bits	Name	Description
[31:11]	-	Reserved, RAZ
[10:0]	SPIID	Sets the SPI ID that the GIC generates when a PMU overflow interrupt occurs. If the value is less than 32, out of range, or not owned on chip for multichip configurations, the register updates to 0 and no internal delivery occurs. Set this field to 0 when the interrupt routes externally to a core that does not receive interrupts directly from the GIC such as a central system control processor. **Note:** The behavior is unpredictable if software attempts to share the same interrupt ID in GICP_IRQCR with an external source using either: • An SPI wire. • The GICD_SETSPI_NSR or GICD_SETSPI_SR registers. Creates a level-triggered interrupt if it is owned on chip. Otherwise it behaves as a normal message-based SPI. In a multichip configuration, the SPIID field must be programmed only to an SPI ID that the chip owns. The relevant GICD_CHIPRn register controls the SPI ownership. We recommend that if these registers are used, then the SPI must not be used for another device either with a wire or as a message-based interrupt.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_IRQCR is accessible only by Secure accesses.

For GIC configurations that support multi view, that is when GICD_CFGID.VIEW == 1, then GICP_IRQCR is accessible only for view 0.

5.11.16 GICP_PIDR2, Peripheral ID2 Register

This register returns byte[2] of the peripheral ID. The GICP_PIDR2 register is part of the set of performance monitoring peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.11 GICP register summary on page 269 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-99: GICP_PIDR2 bit assignments

Table 5-130: GICP_PIDR2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:4]	ArchRev	Identifies the version of the GIC architecture with which the PMU complies: **0x3** GICv3 **0x4** GICv4
[3]	JEDEC	Indicates that a JEDEC-assigned JEP106 identity code is used.
[2:0]	DES_1	Bits[6:4] of the JEP106 identity code. Bits[3:0] of the JEP106 identity code are assigned to GICP_PIDR1[7:4].

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_PIDR2 is accessible only by Secure accesses.

5.12 FMU register summary

The GIC-720AE *Fault Management Unit* (FMU) functions are controlled through registers that are identified with the prefix FMU.

Unless otherwise stated in the accompanying FMU register descriptions:

- Do not modify Reserved register bits.
- Ignore Reserved register bits on reads.

Table 5-131: FMU register summary

Offset	Name	Type	Reset	Width	Description
0x000 + (n × 64)	FMU_ERR<n>FR	RO	0x00000XXX_00000022 for odd error records <n>. 0x00000XXX_00800002 for even error records <n>. Depending on the GIC block type, bits[43:32] are configuration dependent.	64	Error Record <n> Feature Register, n = 0-11
0x008 + (n × 64)	FMU_ERR<n>CTLR	RW	0x3	64	Error Record <n> Control Register, n = 0-11
0x010 + (n × 64)	FMU_ERR<n>STATUS	RW	0x1	64	Error Record <n> Primary Status register, n = 0-11
0xE00	FMU_ERRGSR	RO	0x0	64	Error Group Status Register
0xE10	FMU_ERRIIDR	RO	0x49A0043B	32	Implementation ID Register
0xF00	FMU_SMEN	WO	0x0	32	Safety Mechanism Enable and disable register
0xF04	FMU_SMERR	WO	0x0	32	Safety Mechanism Inject Error register
0xF08	FMU_SMCR	WO	0x0	32	Safety Mechanism Set Criticality Register
0xF0C	FMU_SMWR	WO	0x0	32	Safety Mechanism Page Write Register
0xF10	FMU_SMWDATA	RW	0x0	32	Safety Mechanism Write Data register
0xF14	FMU_SMRD	WO	0x0	32	Safety Mechanism Page Read register
0xF18	FMU_SMRDATA	RO	0x0	32	Safety Mechanism Read Data register
0xF1C	FMU_STATUS	RO	0x0	32	FMU Status register
0xF20	FMU_KEY	RW	0x0	32	FMU Key register
0xF24	FMU_TIMEOUT	RW	0xFFFFFFFF	32	Timeout duration register
0xF28	FMU_ERRUPDATE	WO	0x0	32	Error update register
0xF2C	FMU_FCTLR	RW	0x0	32	Function Control Register
0xFBC	FMU_ERRDEVARCH	RO	0x0	32	FMU Device Architecture register

Offset	Name	Type	Reset	Width	Description
0xFC8	FMU_ERRDEVID	RO	0x0C	32	Device configuration register
0xFD0	FMU_PIDR4	RO	0x04	32	Peripheral ID 4 Register
0xFD4	FMU_PIDR5	RO	0x00	32	Peripheral ID 5 Register
0xFD8	FMU_PIDR6	RO	0x00	32	Peripheral ID 6 Register
0xFDC	FMU_PIDR7	RO	0x00	32	Peripheral ID 7 Register
0xFE0	FMU_PIDR0	RO	0x9A	32	Peripheral ID 0 Register
0xFE4	FMU_PIDR1	RO	0xB4	32	Peripheral ID 1 Register
0xFE8	FMU_PIDR2	RO	0x0B	32	Peripheral ID 2 Register
0xFEC	FMU_PIDR3	RO	0x00	32	Peripheral ID 3 Register
0xFF0	FMU_CIDR0	RO	0x0D	32	Component ID 0 Register
0xFF4	FMU_CIDR1	RO	0xF0	32	Component ID 1 Register
0xFF8	FMU_CIDR2	RO	0x05	32	Component ID 2 Register
0xFFC	FMU_CIDR3	RO	0xB1	32	Component ID 3 Register

5.12.1 FMU_ERR<n>FR, Error Record <n> Feature Register

This register defines which of the common architecturally defined features are implemented and, of the implemented features, which are software programmable. GIC-720AE supports 12 error records, n = 0-11.

The value of n maps to the following error records:

n=0 GICD, critical error record
n=1 GICD, non-critical error record
n=2 Wake Request, critical error record
n=3 Wake Request, non-critical error record
n=4 SPI Collator, critical error record
n=5 SPI Collator, non-critical error record
n=6 GCI, critical error record
n=7 GCI, non-critical error record
n=8 ITS, critical error record
n=9 ITS, non-critical error record
n=10 FMU, critical error record
n=11 FMU, non-critical error record

If a record is not implemented because the block type does not exist in the configuration, then this register is RAZ/WI.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit

Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-100: FMU_ERR<n>FR bit descriptions

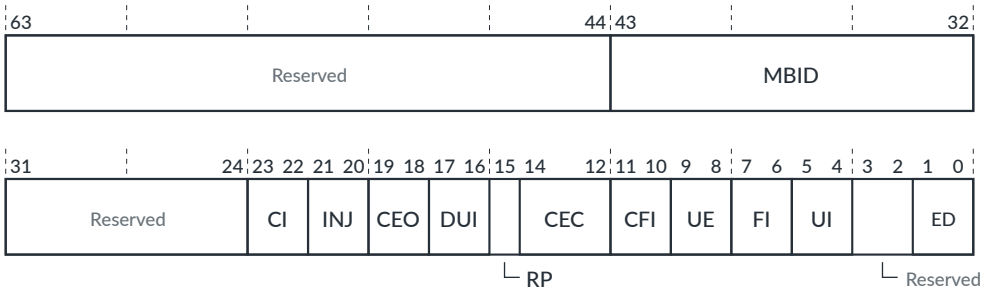


Table 5-132: FMU_ERR<n>FR bit assignments

Bits	Name	Description
[63:44]	-	Reserved, RAZ
[43:32]	MBID	The maximum block ID, which is the number of configured blocks minus 1.
[31:24]	-	Reserved, RAZ
[23:22]	CI	Critical error interrupt. Returns: **0b00** Critical error interrupt is not supported. This value occurs for odd records. **0b10** Critical error interrupt is supported and controllable using FMU_ERR<n>CTLR.CI. This value occurs for even records.
[21:20]	INJ	Fault injection extension. Returns: **0b00** The FMU does not implement the RAS Common Fault Injection Model Extension.
[19:18]	CEO	Corrected error overwrite. This field is RES0 because CEC == 0b000.
[17:16]	DUI	Error recovery interrupt for deferred errors control. Returns: **0b00** The control for enabling error recovery interrupts on deferred errors is not supported.
[15]	RP	Repeat counter. Returns: **0b0** The FMU does not implement the repeat Corrected error counter.
[14:12]	CEC	Corrected error counter. Returns: **0b000** The FMU does not implement the standard Corrected error counter mechanism.

Bits	Name	Description
[11:10]	CFI	Fault handling interrupt for corrected errors control. Returns: **0b00** Fault handling interrupts on corrected errors is not supported.
[9:8]	UE	Indicates whether the in-band error response (External Abort) and associated controls are implemented. Returns: **0b00** In-band error response (External Abort) is not supported.
[7:6]	FI	Fault handling interrupt. Returns: **0b00** Fault handling interrupt is not supported.
[5:4]	UI	Error recovery interrupt for uncorrected errors. Returns: **0b00** Error recovery interrupt is not supported. This value occurs for even records. **0b10** Error recovery interrupt is supported and controllable using FMU_ERR<n>CTLR.UI. This value occurs for odd records.
[3:2]	-	Reserved, RAZ
[1:0]	ED	Error reporting and logging. Returns: **0b10** Error reporting and logging is controllable using FMU_ERR<n>CTLR.ED.

Accessibility

FMU_ERR<n>FR is accessible only by Secure accesses.

5.12.2 FMU_ERR<n>CTLR, Error Record <n> Control Register

For even error records, this register controls whether the FMU can generate a critical error interrupt. For odd error records, this register controls whether the FMU can generate an error recovery interrupt. GIC-720AE supports 12 error records, `n` = 0-11.

The value of `n` maps to the following error records:

n=0 GICD, critical error record
n=1 GICD, non-critical error record
n=2 Wake Request, critical error record
n=3 Wake Request, non-critical error record
n=4 SPI Collator, critical error record
n=5 SPI Collator, non-critical error record
n=6 GCI, critical error record
n=7 GCI, non-critical error record
n=8 ITS, critical error record
n=9 ITS, non-critical error record
n=10 FMU, critical error record
n=11 FMU, non-critical error record

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-101: FMU_ERR<n>CTLR bit assignments

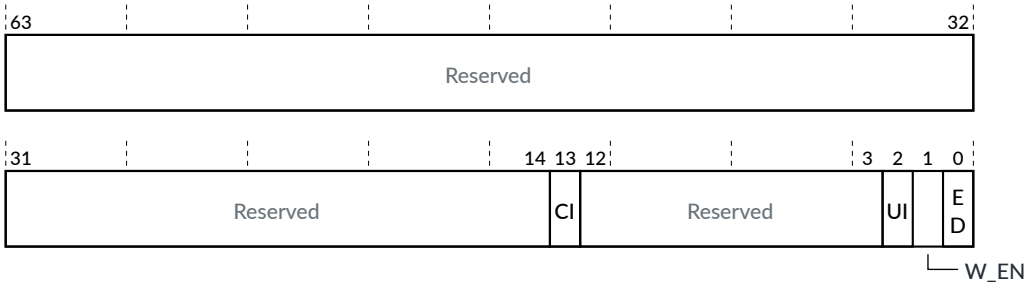


Table 5-133: FMU_ERR<n>CTLR bit descriptions

Bits	Name	Description
[63:14]	-	Reserved, RAZ/WI
[13]	CI	Critical error interrupt enable. For even records: **0** The FMU does not generate critical error interrupts. **1** When ED == 1, the FMU generates a critical error interrupt when a critical error condition occurs. RAZ/WI for odd records.
[12]	-	Reserved, RAZ/WI
[11]	WDUI	Feature not supported, RAZ/WI
[10]	DUI	Feature not supported, RAZ/WI
[9]	WCFI	Feature not supported, RAZ/WI
[8]	CFI	Feature not supported, RAZ/WI
[7]	WUE	Feature not supported, RAZ/WI
[6]	WFI	Feature not supported, RAZ/WI
[5]	WUI	Feature not supported, RAZ/WI
[4]	UE	Feature not supported, RAZ/WI
[3]	FI	Feature not supported, RAZ/WI

Bits	Name	Description
[2]	UI	For odd records, this bit controls whether an *Error Recovery Interrupt* (ERI) is generated for errors that are reported through this error record: **0** The FMU does not generate an error recovery interrupt. **1** When ED == 1, the FMU generates an error recovery interrupt for all errors that this odd error record reports. RAZ/WI for even records.
[1]	W_EN	If a fault causes the error record <n> input signal to assert permanently, software can use this bit to disable that error record input signal: **0** The error record <n> input signal is disabled. **1** The error record <n> input signal is enabled. This setting occurs at reset. The error record <n> input error signals are cr_err_in_* and ncr_err_in_*, where * is ci, its, wake, or spicol. The error record <n> input error signals for the GICD and FMU are not externally accessible.
[0]	ED	Error reporting and logging enable: **0** Error reporting and logging is disabled for this record. **1** Error reporting and logging is enabled for this record. This setting occurs at reset.

Accessibility

FMU_ERR<n>CTLR is accessible only by Secure accesses.

5.12.3 FMU_ERR<n>STATUS, Error Record <n> Primary Status register

This register indicates information relating to the recorded errors in error record <n>, where n = 0-11.

Software can write to this register to clear the error records that FMU_ERRGSR reports.

The value of n maps to the following error records:

n=0 GICD, critical error record
n=1 GICD, non-critical error record
n=2 Wake Request, critical error record
n=3 Wake Request, non-critical error record
n=4 SPI Collator, critical error record
n=5 SPI Collator, non-critical error record
n=6 GCI, critical error record
n=7 GCI, non-critical error record
n=8 ITS, critical error record
n=9 ITS, non-critical error record
n=10 FMU, critical error record
n=11 FMU, non-critical error record

If the error record <n> input error signal asserts or if an error_report message is received, then the Valid (V) bit is set and the other information fields are updated for the new error. The cr_err_in_*

and ncr_err_in_* signals are the error record <n> input error signals, where * is ci, its, wake, or spicol. The error record <n> input error signals for the GICD and FMU are not externally accessible.

If a write to FMU_ERR<n>STATUS causes an access to a powered-down block, then when the FMU sets FMU_STATUS.BUSY=0 it also sets FMU_STATUS.BLKID_PWROFF=1.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit

Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register. This register is reset only by the fmu_reset_n signal.

Usage constraints

- After a write to this register, poll the FMU_STATUS register to ensure that the effect of the write is complete. Until the write takes effect, that is, FMU_STATUS.BUSY==0 then:
  - The corresponding bit of FMU_ERRGSR might still report as 1.
  - Any interrupts caused by this record might still be asserted.
  - Any error that arrives, sets FMU_ERR<n>STATUS in the same way that a new error does.
  - Any read of this register, returns the new error if a new error occurs.
- If software reads V=1, UE=1, and OF=0, software must then write to clear V and UE. If OF=0, then the FMU clears the V and UE bits. However, if V=1, UE=1, and OF=1 and software clears V and UE only, then the FMU does not update FMU_ERR<n>STATUS, in accordance with RAS architecture v1.1, leaving V=1, UE=1, and OF=1. Therefore, after attempting a clear, another read is required to determine if V clears. If it did not clear and OF is now 1, clear again with OF set to 1.
- If software reads V=1, UE=1, and OF=1 it is not possible to know the exact number of errors that occurred for the reported BLKID and PROTID, only that it was at least two errors.
- If software reads V=1, UE=1, and OFB=1, it must clear the error normally, and the fault collator in the GIC block that caused the OFB, resends the PROTID for the second error on receiving the clear message for the first error. As a result of the fault collator barrier behavior, resend of the second error is guaranteed to have been sent by the fault collator (and received in the FMU) before the FMU receives the clear acknowledge for the first error. Receipt of the clear acknowledge is indicated by polling FMU_STATUS.BUSY==0.
- If software reads V=1, UE=1, and OFX=1, it must clear the error normally, and it must then trigger an error resend by writing the record pair ID to FMU_ERRUPDATE. This write causes V, UE, OF, BLKID, and IERR to update with any outstanding errors from other BLKIDs. UE is a copy of the V bit, so the values written to V and UE must be the same.
- In the FMU, one BLKID corresponds to one fault collator in the GIC system. If an FMU register access to a fault collator requires that one or more errors are sent, the fault collator barrier behavior ensures that updates to the corresponding FMU_ERR<n>STATUS register occur

before FMU_STATUS.BUSY==0. The fault collator barrier behavior applies both to real errors that actual faults cause and to inserted errors that writes to FMU_SMERR cause.

- The V bit can only be cleared, when the error packet from the block with the error has been received, that is, IERR is nonzero, and when either:
  - The error wire has been received from the block with the error, that is, W=1.
  - The error wire for the error record is disabled in the FMU, that is, FMU_ERR<n>CTLR.W_EN=0.

  To enable or disable error wire reporting in a block, software must use PROTID 255 for that block. See 6.1.6.4 Enabling or disabling both error signals on a block on page 351 for more information.

Bit descriptions

Figure 5-102: FMU_ERR<n>STATUS bit assignments

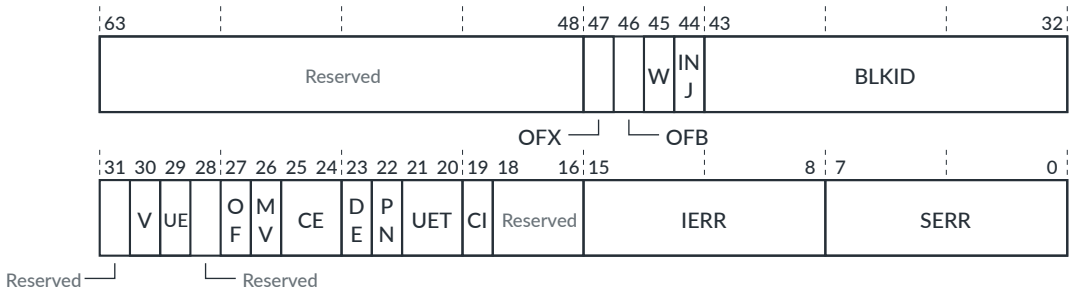


Table 5-134: FMU_ERR<n>STATUS bit descriptions

Bits	Name	Description	Type
[63:48]	-	Reserved, RAZ	-
[47]	OFX	When 1, indicates that one or more errors have been received from one or more different BLKIDs, compared to the block ID that BLKID reports.	RO
[46]	OFB	When 1, indicates that the reported BLKID has one or more errors for a different PROTID compared to the PROTID that the IERR field reports. Software can clear this bit by setting V = 0.	RO
[45]	W	When 1, indicates that the error record <n> input error signal is asserted.	RO
[44]	INJ	When 1, indicates that FMU_SMERR injected the reported error. This bit is not valid when FMU_ERR<n>STATUS.V == 0.	RO
[43:32]	BLKID	This field indicates the ID of the component block that is reporting an error. This field is valid only when FMU_ERR<n>STATUS.V==1. The BLKID that is used to access a GCI does not change, even if processors are removed from a pre-configured GIC. See A.1 Removing cores from a preconfigured GIC on page 361. When BLKID is not known, this field becomes 0. The BLKID might be unknown when software clears V or when the error wire for record <n> is received.	RO
[31]	-	Reserved, RAZ	-

Bits	Name	Description	Type
[30]	V	Indicates if this register is valid: **0** FMU_ERR<n>STATUS is not valid. **1** FMU_ERR<n>STATUS is valid. One or more errors are recorded. Write 1 to clear. When clearing this bit, FMU_ERR<n>STATUS.UE must also be cleared.	RW
[29]	UE	Uncorrected error bit. This bit value is the same as the V bit, because errors are always reported as uncorrected. Software can read the IERR field to determine if the error is due to a RAM SEC error.	RW
[28]	-	Reserved, RAZ	-
[27]	OF	Reported PROTID has overflowed: **0** No overflow for the current PROTID. **1** More than one error is recorded against the current PROTID. To clear this bit, software writes OF = 1, V = 1, UE = 1 to this register.	RW
[26]	MV	Miscellaneous registers valid bit is not supported, RAZ/WI.	-
[25:24]	CE	Corrected error field is not supported, RAZ/WI.	-
[23]	DE	Deferred error bit is not supported, RAZ/WI.	-
[22]	PN	Poison bit is not supported, RAZ/WI.	-
[21:20]	UET	Uncorrected error type. Returns: **0b11** Uncorrected error, Signaled or Recoverable error (UER). This field is not valid and reads as zero if either of the following conditions are true: • FMU_ERR<n>STATUS.V == 0 • FMU_ERR<n>STATUS.UE == 0	RO
[19]	CI	Indicates whether a critical error condition has been recorded: **0** No critical error condition. **1** Critical error condition recorded. This bit is not valid and reads as zero when FMU_ERR<n>STATUS.V == 0. For non-critical records (odd IDs), this field is always 0. For critical records (even IDs), this bit reports the same value as V and it does not need to be written when clearing V.	RW
[18:16]	-	Reserved, RAZ	-
[15:8]	IERR	Implementation-defined error code. Contains the *Protection Mechanism ID* (PROTID), which indicates the protection mechanism reporting the error. If FMU_ERR<n>STATUS.V == 0, this field is not valid and reads as zero. When V=1 but the PROTID is not yet known, this field is set to 0.	RO
[7:0]	SERR	Architecturally defined primary error code. If an error occurs, this field returns: **0x01** Implementation-defined error.	RO

Accessibility

FMU_ERR<n>STATUS is accessible only by Secure accesses.

5.12.4 FMU_ERRGSR, Error Group Status Register

This register shows the status of all FMU_ERR<n>STATUS.V bits.

Configurations

This register is available in all configurations.

Attributes

Width 64-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-103: FMU_ERRGSR bit assignments

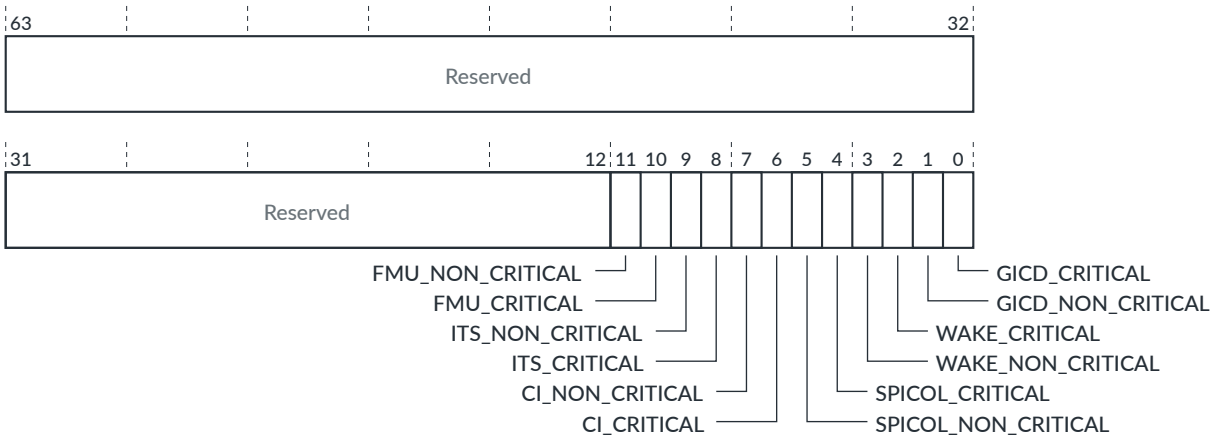


Table 5-135: FMU_ERRGSR bit descriptions

Bits	Name	Description
[63:12]	-	Reserved, RAZ
[11]	FMU_NON_CRITICAL	Returns the status of the FMU non-critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[10]	FMU_CRITICAL	Returns the status of the FMU critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[9]	ITS_NON_CRITICAL	Returns the status of the combined ITS hardware non-critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.

Bits	Name	Description
[8]	ITS_CRITICAL	Returns the status of the combined ITS hardware critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[7]	CI_NON_CRITICAL	Returns the status of the combined CI hardware non-critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[6]	CI_CRITICAL	Returns the status of the combined CI hardware critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[5]	SPICOL_NON_CRITICAL	Returns the status of the combined SPI Collator hardware non-critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[4]	SPICOL_CRITICAL	Returns the status of the combined SPI Collator hardware critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[3]	WAKE_NON_CRITICAL	Returns the status of the Wake Request non-critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[2]	WAKE_CRITICAL	Returns the status of the Wake Request critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[1]	GICD_NON_CRITICAL	Returns the status of the GICD non-critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.
[0]	GICD_CRITICAL	Returns the status of the GICD critical error record: **0** The error record is not reporting any errors. **1** The error record is reporting one or more errors.

Accessibility

FMU_ERRGSR is accessible only by Secure accesses.

5.12.5 FMU_ERRIIDR, FMU Implementer Identification Register

This register provides information about the implementer and revision of the FMU.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-104: FMU_ERRIIDR bit assignments

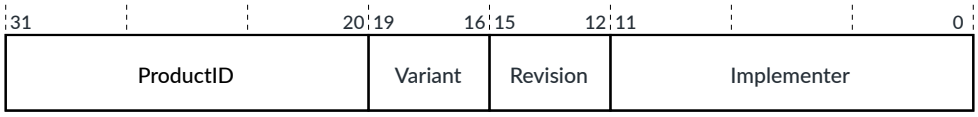


Table 5-136: FMU_ERRIIDR bit descriptions

Bits	Name	Description
[31:20]	ProductID	Indicates the part number of the component: **0x49A** GIC-720AE FMU
[19:16]	Variant	Indicates the major revision, or variant, of the FMU: **0x0** r0
[15:12]	Revision	Indicates the minor revision of the FMU: **0x0** p0
[11:0]	Implementer	Identifies the implementer: **0x43B** Arm

5.12.6 FMU_SMEN, Safety Mechanism Enable register

This register enables or disables particular protection mechanisms inside a specified GIC block. At reset, the GIC enables all the protection mechanisms. We recommend that software does not disable any protection mechanisms.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

- After a write to this register, poll FMU_STATUS.BUSY to ensure that the effect of the write is complete.
- Do not write to FMU_SMEN and enable or disable a protection mechanism that corresponds to a powered-off block. See Power management on page 355.

Note

If a block is powered-off and then powered-on again, the enabled state of the protection mechanism returns to the default reset state.

Bit descriptions

Figure 5-105: FMU_SMEN bit assignments

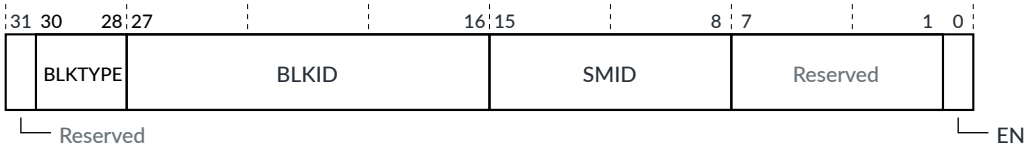


Table 5-137: FMU_SMEN bit descriptions

Bits	Name	Description
[31]	-	Reserved, RAZ
[30:28]	BLKTYPE	Block type identifier: **0** GICD **1** Wake Request **2** SPI Collator **3** *GIC Cluster Interface* (GCI) **4** ITS **5** FMU
[27:16]	BLKID	Block identifier. The permitted values are: • 0 when BLKTYPE == 0, 1, or 5. • 0-FMU_ERR4FR.MBID when BLKTYPE == 2. • 0-FMU_ERR6FR.MBID when BLKTYPE == 3. The BLKID that is used to access a GCI does not change, even if processors are removed from a pre-configured GIC. See A.1 Removing cores from a preconfigured GIC on page 361. • 0-FMU_ERR8FR.MBID when BLKTYPE == 4.

Bits	Name	Description
[15:8]	SMID	Protection mechanism identifier. See 6.1.6 Protection mechanism IDs on page 336 for the valid protection mechanism ID encodings for each BLKTYPE. If software writes SMID=255 and EN=0, this combination disables error signal reporting for the block that BLKTYPE and BLKID select. If software writes SMID=255 and EN=1, this combination enables error signal reporting for the block that BLKTYPE and BLKID select. Also, software can use the SMID=255 value to request a resend of errors from a specific block.
[7:1]	-	Reserved, RAZ
[0]	EN	Enables or disables a protection mechanism: **0** Disables a protection mechanism with the ID that SMID contains. **1** Enables a protection mechanism with the ID that SMID contains.

Accessibility

FMU_SMEN is accessible only by Secure accesses.

5.12.7 FMU_SMERR, Safety Mechanism Inject Error register

This register injects one error into the specified protection mechanism inside a GIC block. Writes to this register cause an FMU_CTRL_ACCESS message to be sent with err_insert=1.

By using this register, the system integrator must check a single protection mechanism from each block at Cold reset, to insert an error and check that the error wire and AXI5-Stream packet reporting to the FMU occurs. When a block type can have multiple instances such as an ITS or GCI, then the system integrator must check all instances of those blocks.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

- After a write to this register, poll FMU_STATUS.BUSY to ensure that the effect of the write is complete.
- Do not write to FMU_SMERR and inject an error that corresponds to a powered-off block. See Power management on page 355.

Bit descriptions

Figure 5-106: FMU_SMERR bit assignments

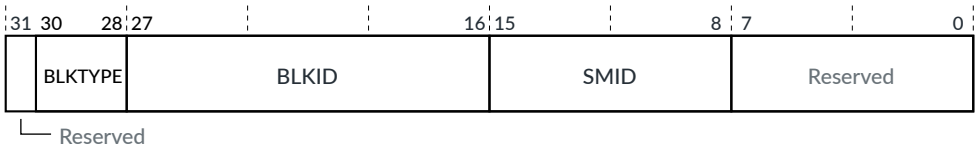


Table 5-138: FMU_SMERR bit descriptions

Bits	Name	Description
[31]	-	Reserved, RAZ
[30:28]	BLKTYPE	Block type identifier: **0** GICD **1** Wake Request **2** SPI Collator **3** *GIC Cluster Interface* (GCI) **4** ITS **5** FMU
[27:16]	BLKID	Block identifier. The permitted values are: • 0 when BLKTYPE == 0, 1, or 5. • 0-FMU_ERR4FR.MBID when BLKTYPE == 2. • 0-FMU_ERR6FR.MBID when BLKTYPE == 3. The BLKID that is used to access a GCI does not change, even if processors are removed from a pre-configured GIC. See A.1 Removing cores from a preconfigured GIC on page 361. • 0-FMU_ERR8FR.MBID when BLKTYPE == 4.
[15:8]	SMID	Protection mechanism identifier. See 6.1.6 Protection mechanism IDs on page 336 for the valid protection mechanism ID encodings for each BLKTYPE. Software can use the SMID=255 value to request a resend of errors from a specific block.
[7:0]	-	Reserved, RAZ

Accessibility

FMU_SMERR is accessible only by Secure accesses.

5.12.8 FMU_SMCR, Safety Mechanism Set Criticality Register

This register sets the protection mechanism criticality. When the FMU receives a write access to this register then it sends an FMU_CTRL_ACCESS message to the fault collators that reside in the other GIC components.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

- After a write to this register, poll FMU_STATUS.BUSY to ensure that the effect of the write is complete.
- Do not write to FMU_SMCR that corresponds to a powered-off block. See Power management on page 355.

Bit description

Figure 5-107: FMU_SMCR bit assignments

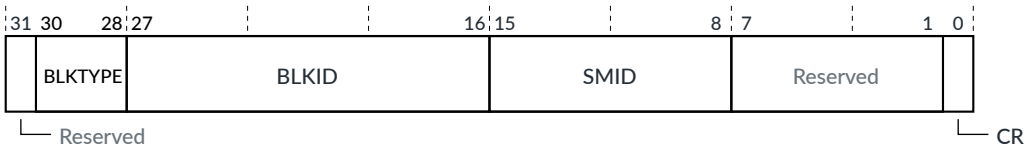


Table 5-139: FMU_SMCR bit descriptions

Bits	Name	Description
[31]	-	Reserved, RAZ
[30:28]	BLKTYPE	Block type identifier: **0** GICD **1** Wake Request **2** SPI Collator **3** *GIC Cluster Interface* (GCI) **4** ITS **5** FMU
[27:16]	BLKID	Block identifier. The permitted values are: • 0 when BLKTYPE == 0, 1, or 5. • 0-FMU_ERR4FR.MBID when BLKTYPE == 2. • 0-FMU_ERR6FR.MBID when BLKTYPE == 3. The BLKID that is used to access a GCI does not change, even if processors are removed from a pre-configured GIC. See A.1 Removing cores from a preconfigured GIC on page 361. • 0-FMU_ERR8FR.MBID when BLKTYPE == 4.
[15:8]	SMID	Safety Mechanism identifier. See 6.1.6 Protection mechanism IDs on page 336 for the valid protection mechanism ID encodings for each BLKTYPE. Software can use the SMID=255 value to request a resend of errors from a specific block.
[7:1]	-	Reserved, RAZ

Bits	Name	Description
[0]	CR	Sets the criticality of a protection mechanism: **0** Sets the protection mechanism as a non-critical error. As the GIC exits reset, it applies this setting to the RAM SEC errors, that is, the SM_SEC* protection mechanisms. **1** Sets the protection mechanism as a critical error. As the GIC exits reset, it applies this setting to all protection mechanisms except for the RAM SEC protection mechanisms.

Accessibility

FMU_SMCR is accessible only by Secure accesses.

5.12.9 FMU_SMWR, Safety Mechanism Page Write Register

This register performs a page write access that is then followed by a page read access. When the FMU receives a write access to this register then it sends an FMU_PAGE_ACCESS message to the fault collators that reside in the other GIC components.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

- Before software writes to this register, it must write to FMU_SMWDATA.
- After a write to this register, poll FMU_STATUS.BUSY to ensure that the effect of the write is complete.
- Do not write to FMU_SMWR that corresponds to a powered-off block. See Power management on page 355.

Bit description

Figure 5-108: FMU_SMWR bit assignments

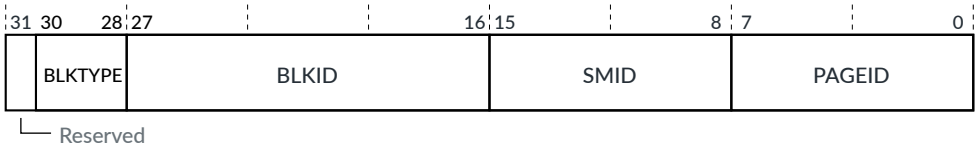


Table 5-140: FMU_SMWR bit descriptions

Bits	Name	Description
[31]	-	Reserved, RAZ

Bits	Name	Description
[30:28]	BLKTYPE	Block type identifier: **0** GICD **1** Wake Request **2** SPI Collator **3** *GIC Cluster Interface* (GCI) **4** ITS **5** FMU
[27:16]	BLKID	Block identifier. The permitted values are: • 0 when BLKTYPE == 0, 1, or 5. • 0-FMU_ERR4FR.MBID when BLKTYPE == 2. • 0-FMU_ERR6FR.MBID when BLKTYPE == 3. The BLKID that is used to access a GCI does not change, even if processors are removed from a pre-configured GIC. See A.1 Removing cores from a preconfigured GIC on page 361. • 0-FMU_ERR8FR.MBID when BLKTYPE == 4.
[15:8]	SMID	Safety Mechanism identifier. See 6.1.6 Protection mechanism IDs on page 336 for the valid protection mechanism ID encodings for each BLKTYPE. Software can use the SMID=255 value to request a resend of errors from a specific block.
[7:0]	PAGEID	The ID of the page to write to. The maximum value of this field is: • 0, for blocks that do not have interrupt protection or AXI5-Stream protection. • 1, for blocks that have AXI5-Stream protection. • 2, for blocks that have interrupt protection. • 3, for blocks that have CPU interface protection. After software writes to this register, the FMU sends an FMU_PAGE_ACCESS message to the fault collators with the following data: • `write_valid == 1` • `write_page == PAGEID` • `read_valid == 0` • `read_page == 0` The FMU stores the read data into FMU_SMRDATA.

Accessibility

FMU_SMWR is accessible only by Secure accesses.

5.12.10 FMU_SMWDATA, Safety Mechanism Write Data register

This register contains the data that is written during a page write access.

Writes to this register does not set FMU_STATUS.BUSY = 1 or the sending of AXI5-Stream messages.

The format of the register depends on which page is being written, and the number of pages available depends on the protection mechanism:

Page 0 Available for all protection mechanisms.
Page 1 Available for:

- AXI5-Stream protection, AXI5-Stream cross-chip protection, ACE5-Lite cross-chip protection, and *CPU interface* (CPUIF) protection.
- Interrupt protection.

Page 2 Available for interrupt protection, AXI5-Stream protection, and CPUIF protection.
Page 3-4 Available for CPUIF protection.

Configurations

This register is available in all configurations.

If a GIC configuration does not include a particular protection type, then software can still perform page accesses to the corresponding PROTIDs. In this case, when using FMU_SMWR and FMU_SMWDATA to perform a write:

- Writes to page 0 are visible for subsequent reads using FMU_SMRD and FMU_SMRDATA.
- Writes to page 1 and higher are ignored, and reading them back using FMU_SMRD and FMU_SMRDATA returns zeros.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

Software must program this register before each write to FMU_SMWR.

Bit descriptions

The bit descriptions depend on the page that is written. Some protection mechanisms provide multiple pages for the control and status information.

The following table shows the page 0 writes for all protection mechanisms.

Table 5-141: FMU_SMWDATA bit descriptions for page 0 (FMU_SMWR.PAGEID==0)

Bits	Name	Description
[31:5]	-	Reserved
[4]	inserted	Insert, or clear, an error for testing purposes: **0** Error is not inserted. **1** Insert, or clear, an error.
[3]	-	Reserved

Bits	Name	Description
[2]	critical	Sets the criticality of a protection mechanism: **0** The protection mechanism, with the ID that FMU_SMWR.SMID wrote, is set as a non-critical error. As the GIC exits reset, it applies this setting to the RAM SEC protection mechanisms, that is, the SM_SEC* protection mechanisms. **1** The protection mechanism, with the ID that FMU_SMWR.SMID wrote, is set as a critical error. As the GIC exits reset, it applies this setting to all protection mechanisms except for the RAM SEC protection mechanisms.
[1]	enable	Enables or disables a protection mechanism: **0** Disables a protection mechanism with the ID that FMU_SMWR.SMID wrote. **1** Enables a protection mechanism with the ID that FMU_SMWR.SMID wrote.
[0]	-	Reserved

Page 1 writes for AXI5-Stream protection and CPUIF protection

For AXI5-Stream interface and CPUIF protection, this write sets the timeout behavior for CRC protection. CRC protection schedules a ping packet to be sent after any mission or FMU packet, except the PowerdownAck packet. After a ping packet is received, the recipient must respond with a ping acknowledge packet.

Table 5-142: FMU_SMWDATA bit descriptions for page 1 (FMU_SMWR.PAGEID==1)

Bits	Name	Description
[31:16]	SendTime	Sets the duration when it becomes a high priority for the block to send a ping or a ping acknowledge packet: • 4 × (SendTime + 1) − 1 cycles.
[15:0]	ErrTime	Sets the duration when the block detects a timeout error because a ping or a ping acknowledge packet is missing: • 64 × (ErrTime + 1) − 1 cycles. We recommend that the time to error is at least 4 times longer than the time to send a ping or a ping acknowledge packet. However, interconnect delays or the different frequencies of both domains might require this recommendation to be longer.

Page 1 writes for AXI5-Stream cross-chip protection

If GICD_CFGID.ACE_CC == 0, sets the timeout behavior for the CRC protection that an AXI5-Stream cross-chip interface uses.

Table 5-143: FMU_SMWDATA bit descriptions for page 1 (FMU_SMWR.PAGEID==1)

Bits	Name	Description
[31:16]	ReadyErrTime	Sets the duration when the block detects a timeout error, because the transmitter does not receive a TREADY response: • 12 × ReadyErrTime + 1 to 16 × ReadyErrTime cycles.
[15:0]	CRCErrTime	Sets the duration when the block detects a timeout error, because a CRC request packet or a CRC acknowledge packet is missing: • 56 × CRCErrTime + 1 to 64 × CRCErrTime cycles.

Page 1 writes for ACE5-Lite cross-chip protection

If GICD_CFGID.ACE_CC == 1, sets the timeout behavior for the CRC protection that an ACE5-Lite cross-chip interface uses.

Table 5-144: FMU_SMWDATA bit descriptions for page 1 (FMU_SMWR.PAGEID==1)

Bits	Name	Description
[31:16]	ReadyErrTime	Sets the duration when the block detects a timeout error, because the manager does not receive an AWREADY response: • 12 × ReadyErrTime + 1 to 16 × ReadyErrTime cycles.
[15:0]	CRCErrTime	Sets the duration when the block detects a timeout error, because a CRC request packet or a CRC acknowledge packet is missing: • 56 × CRCErrTime + 1 to 64 × CRCErrTime cycles.

Page 1 writes for interrupt protection

If a GIC configuration does not enable interrupt protection for the block being accessed, the GIC ignores writes to interrupt protection page 1 and FMU_STATUS.PROTID_ERR returns `0b0`. The `spi_protection_type`, `rlt_spi_protection_type`, and `ppi_protection_type` parameters control whether a configuration supports interrupt protection.

Table 5-145: FMU_SMWDATA bit descriptions for page 1 (FMU_SMWR.PAGEID==1)

Bits	Name	Description
[31:16]	-	Reserved
[15]	Enable	Enable interrupt protection: **0** Disable interrupt protection for the interrupt that INTID selects. **1** Enable interrupt protection for the interrupt that INTID selects.
[14]	EnableTransient	Enable transient protection: **0** Disable transient protection for the interrupt that INTID selects. **1** Enable transient protection for the interrupt that INTID selects.
[13:10]	-	Reserved
[9:0]	INTID_index	Selects the interrupt ID of the PPI or SPI to write to. This value is the wire index for that block of protected interrupts.

Page 2 writes for AXI5-Stream protection and CPUIF protection

The following table shows the page 2 writes that clear the CRC timeout and CRC checksum errors for AXI5-Stream protection and CPUIF protection.

Table 5-146: FMU_SMWDATA bit descriptions for page 2 (FMU_SMWR.PAGEID==2)

Bits	Name	Description
[31:2]	-	Reserved
[1]	clr_crc_timeout_err	Set to 1, to clear a CRC timeout error.
[0]	clr_crc_chksum_err	Set to 1, to clear a CRC checksum error.

Page 2 writes for interrupt protection

If a GIC configuration does not enable interrupt protection for the block being accessed, the GIC ignores writes to interrupt protection page 2 and FMU_STATUS.PROTID_ERR returns `0b0`. The `spi_protection_type`, `rlt_spi_protection_type`, and `ppi_protection_type` parameters control whether a configuration supports interrupt protection.

Table 5-147: FMU_SMWDATA bit descriptions for page 2 (FMU_SMWR.PAGEID==2)

Bits	Name	Description
[31]	clear_err_INTID_valid	Clears the error valid bit: **0** No change in the error valid bit. **1** Clear the error_INTID_valid bit. The clear is successful only if the value of clear_error_overflow is the same as FMU_SMRDATA.error_overflow, in interrupt protection page 2, when the clear is received.
[30]	clear_error_overflow	Clears the error overflow bit: **0** No change in the error overflow bit. **1** Clear the error overflow bit for the reported INTID that FMU_SMRDATA reports for page 2 for interrupt protection. The clear of error_overflow is successful only when software sets clear_err_INTID_valid=1 and clear_error_overflow=1 in the same write access.
[29:0]	-	Reserved

Page 3 writes for CPUIF protection

In the following table, `NUM_EXT_ERR_IF` is a build-time option of the CPUIF protection block. See 6.4 External error inputs on page 357 for more information.

Table 5-148: FMU_SMWDATA bit descriptions for page 3 (FMU_SMWR.PAGEID==3)

Bits	Name	Description
[`NUM_EXT_ERR_IF` + 23:24]	ext_err	Set to 1, to clear an error on an external interface on a CPUIF protection block.
[23]	clr_lockstep_err	Set to 1, to clear a lock-step error.
[22]	clr_cpu_parity_err	Set to 1, to clear a CPU parity error.
[21]	clr_ci_parity_err	Set to 1, to clear a GCI parity error.
[20]	clr_axit_crc_err	Set to 1, to clear an AXI5-Stream CRC error.
[19]	clr_dft_err	Set to 1, to clear a DFT error.
[18]	clr_qch_err	Set to 1, to clear a Q-Channel error.
[17]	clr_clk_err	Set to 1, to clear a clock error.
[16]	clr_reset_err	Set to 1, to clear a reset error.
[`NUM_EXT_ERR_IF` + 7:8]	en_ext_err	Set to 1, if you want an external interface error to contribute to a CPUIF protection fault.
[7]	en_lockstep_err	Set to 1, if you want a lock-step error to contribute to a CPUIF protection fault.
[6]	en_cpu_parity_err	Set to 1, if you want a CPU parity error to contribute to a CPUIF protection fault.
[5]	en_ci_parity_err	Set to 1, if you want a GCI parity error to contribute to a CPUIF protection fault.

Bits	Name	Description
[4]	en_axit_crc_err	Set to 1, if you want an AXI5-Stream CRC error to contribute to a CPUIF protection fault.
[3]	en_dft_err	Set to 1, if you want a DFT error to contribute to a CPUIF protection fault.
[2]	en_qch_err	Set to 1, if you want a Q-Channel error to contribute to a CPUIF protection fault.
[1]	en_clk_err	Set to 1, if you want a clock error to contribute to a CPUIF protection fault.
[0]	en_reset_err	Set to 1, if you want a reset error to contribute to a CPUIF protection fault.

Page 4 writes for CPUIF protection

The following table shows the page 4 writes for a CPUIF protection.

Table 5-149: FMU_SMWDATA bit descriptions for page 4 (FMU_SMWR.PAGEID==4)

Bits	Name	Description	Type
[31]	deadlock_error_seen	When set to 1, it indicates the occurrence of a deadlock error. Write one to clear this bit.	R/W1C
[30]	deadlock_detection_enable	Set to 1, to enable deadlock detection. We recommend that deadlock detection timeout is not disabled because it could cause transactions to back up during error conditions, which impacts other processors and complicates the recovery.	RW
[29]	deadlock_correction_enable	Set to 1, to enable deadlock correction.	RW
[28:20]	-	Reserved	-
[19:5]	iritready_timeout	Sets the value of the iritready signal timeout to 8192 × (iritready_timeout + 1) − 1 cycles. Times out if the processor does not respond. However, this timeout is not on the interface, so there must be two transactions stuck ahead before it times out. These are protected by lock-stepping.	RW
[4:0]	interbeat_timeout	Sets the value of the interbeat timeout to 256 × (interbeat_timeout + 1) − 1 cycles. Used for timing between beats of multi-beat transactions.	RW

Accessibility

FMU_SMWDATA is accessible only by Secure accesses.

5.12.11 FMU_SMRD, Safety Mechanism Page Read Register

This register performs a page read access. When the FMU receives a write access to this register, it sends an FMU_PAGE_ACCESS message to the fault collator that the BLKTYPE and BLKID fields select.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit

Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

- After a write to this register, poll FMU_STATUS.BUSY to ensure that the effect of the write is complete.
- Do not write to FMU_SMRD that corresponds to a powered-off block. See Power management on page 355.

Bit description

Figure 5-109: FMU_SMRD bit assignments

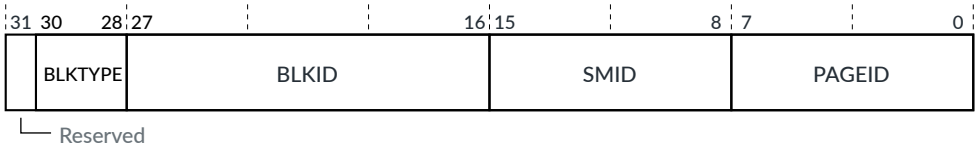


Table 5-150: FMU_SMRD bit descriptions

Bits	Name	Description
[31]	-	Reserved, RAZ
[30:28]	BLKTYPE	Block type identifier: **0** GICD **1** Wake Request **2** SPI Collator **3** *GIC Cluster Interface* (GCI) **4** ITS **5** FMU
[27:16]	BLKID	Block identifier. The permitted values are: • 0 when BLKTYPE == 0, 1, or 5. • 0-FMU_ERR4FR.MBID when BLKTYPE == 2. • 0-FMU_ERR6FR.MBID when BLKTYPE == 3. The BLKID that is used to access a GCI does not change, even if processors are removed from a pre-configured GIC. See A.1 Removing cores from a preconfigured GIC on page 361. • 0-FMU_ERR8FR.MBID when BLKTYPE == 4.
[15:8]	SMID	Protection mechanism identifier. See 6.1.6 Protection mechanism IDs on page 336 for protection mechanism ID encodings.

Bits	Name	Description
[7:0]	PAGEID	The ID of the page to read from. The maximum value of this field is: • 0, for blocks that do not have interrupt protection or AXI5-Stream protection. • 1, for blocks that have AXI5-Stream protection. • 2, for blocks that have interrupt protection. After software writes to this register, the FMU sends an FMU_PAGE_ACCESS message with the following data: • `read_valid == 1` • `read_page == PAGEID` • `write_valid == 0` • `write_page == 0`

Accessibility

FMU_SMRD is accessible only by Secure accesses.

5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register

This register contains the data that is read during a page read access.

The format of the register depends on which page is being read, and the number of pages available depends on the protection mechanism:

Page 0 Available for all protection mechanisms.

Page 1 Available for:

- AXI5-Stream protection, AXI5-Stream cross-chip protection, ACE5-Lite cross-chip protection, and *CPU interface* (CPUIF) protection.
- Interrupt protection.

Page 2 Available for interrupt protection, AXI5-Stream protection, and CPUIF protection.

Page 3-4 Available for CPUIF protection.

Configurations

This register is available in all configurations.

If a GIC configuration does not include a particular protection type, then software can still perform page accesses to the corresponding PROTIDs. Reads from page 0 return valid data, but reads from page 1 and higher return zeros.

Attributes

Width 32-bit

Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

If FMU_STATUS.PROTID_ERR == 1, then software must ignore the value of this register.

Bit descriptions

The bit descriptions depend on the page that is read. Some protection mechanisms provide multiple pages for the status information.

The following table shows the page 0 reads for all protection mechanisms.

Table 5-151: FMU_SMRDATA bit descriptions for page 0 (FMU_SMRD.PAGEID==0)

Bits	Name	Description
[31:5]	-	Reserved
[4]	inserted	Returns the error insertion status: **0** No error inserted. **1** Error inserted.
[3]	-	Reserved
[2]	critical	Returns the protection mechanism criticality: **0** The protection mechanism reports a non-critical error. **1** The protection mechanism reports a critical error.
[1]	enable	For FMU_SMRD.SMID ≠ 255, this bit returns whether the protection mechanism is enabled: **0** The protection mechanism with the ID that SMID contains is not enabled. **1** The protection mechanism with the ID that SMID contains is enabled. For FMU_SMRD.SMID = 255, this bit returns whether the error wire outputs for that block are enabled: **0** Both error wire outputs for that block are not enabled. FMU_SMRD.[BLKTYPE|BLKID] selects the block. **1** Both error wire outputs for that block are enabled. FMU_SMRD.[BLKTYPE|BLKID] selects the block.
[0]	-	Reserved

Page 1 reads for AXI5-Stream protection and CPUIF protection

Returns the timeout behavior for the CRC protection that the AXI5-Stream interfaces and CPUIFs use.

Table 5-152: FMU_SMRDATA bit descriptions for page 1 (FMU_SMRD.PAGEID==1)

Bits	Name	Description
[31:16]	SendTime	Returns the duration when it becomes a high priority for the block to send a ping or a ping acknowledge packet: • 4 × (SendTime + 1) − 1 cycles.

Bits	Name	Description
[15:0]	ErrTime	Returns the duration when the block detects a timeout error, because a ping or a ping acknowledge packet is missing: • 64 × (ErrTime + 1) − 1 cycles. We recommend that the time to error is at least 4 times longer than the time to send a ping or a ping acknowledge packet. However, interconnect delays or the different frequencies of both domains might require this recommendation to be longer.

Page 1 reads for AXI5-Stream cross-chip protection

If GICD_CFGID.ACE_CC == 0, returns the timeout behavior for the CRC protection that an AXI5-Stream cross-chip interface uses.

Table 5-153: FMU_SMRDATA bit descriptions for page 1 (FMU_SMRD.PAGEID==1)

Bits	Name	Description
[31:16]	ReadyErrTime	Returns the duration for the TREADY timeout. The value is in the range of 12 × ReadyErrTime + 1 to 16 × ReadyErrTime cycles, and depends on what was written using FMU_SMWDATA.
[15:0]	CRCErrTime	Returns the duration when the block detects a timeout error, because a CRC request packet or a CRC acknowledge packet is missing. The value is in the range of 56 × CRCErrTime + 1 to 64 × CRCErrTime cycles, and depends on what was written using FMU_SMWDATA.

Page 1 reads for ACE5-Lite cross-chip protection

If GICD_CFGID.ACE_CC == 1, returns the timeout behavior for the CRC protection that an ACE5-Lite cross-chip interface uses.

Table 5-154: FMU_SMRDATA bit descriptions for page 1 (FMU_SMRD.PAGEID==1)

Bits	Name	Description
[31:16]	ReadyErrTime	Returns the duration for the AWREADY timeout. The value is in the range of 12 × ReadyErrTime + 1 to 16 × ReadyErrTime cycles, and depends on what was written using FMU_SMWDATA.
[15:0]	CRCErrTime	Returns the duration when the block detects a timeout error, because a CRC request packet or a CRC acknowledge packet is missing. The value is in the range of 56 × CRCErrTime + 1 to 64 × CRCErrTime cycles, and depends on what was written using FMU_SMWDATA.

Page 1 reads for interrupt protection

If a GIC configuration does not enable interrupt protection for the block being accessed, FMU_SMRDATA interrupt protection page 1 returns zero and FMU_STATUS.PROTID_ERR returns `0b0`. The `spi_protection_type`, `rlt_spi_protection_type`, and `ppi_protection_type` parameters control whether a configuration supports interrupt protection.

Table 5-155: FMU_SMRDATA bit descriptions for page 1 (FMU_SMRD.PAGEID==1)

Bits	Name	Description
[31:30]	-	Reserved

Bits	Name	Description
[29]	DetectionPaused	Detection is paused for the INTID being read. After BIST or after programming of an INTID, this bit is set to prevent false error detection and automatically clears when detection is re-enabled. A permanent fault on an interrupt wire or its check wire might prevent this bit from clearing for the INTID being read.
[28]	BISTBusy	This bit is set after reset for a fixed duration, while the GIC performs BIST for the interrupt protection.
[27:16]	-	Reserved
[15]	Enable	Returns the interrupt protection status: **0** Interrupt protection is disabled for the interrupt that INTID selects. **1** Interrupt protection is enabled for the interrupt that INTID selects.
[14]	EnableTransient	Returns the transient protection status: **0** Transient protection is disabled for the interrupt that INTID selects. **1** Transient protection is enabled for the interrupt that INTID selects.
[13:10]	-	Reserved
[9:0]	INTID_index	Returns the interrupt ID of the PPI or SPI that is read. This value is the wire index for that block of protected interrupts.

Page 2 reads for AXI5-Stream protection and CPUIF protection

The following table shows the page 2 reads that return the CRC timeout and CRC checksum errors for AXI5-Stream protection and CPUIF protection.

Table 5-156: FMU_SMRDATA bit descriptions for page 2 (FMU_SMRD.PAGEID==2)

Bits	Name	Description
[31:2]	-	Reserved
[1]	crc_timeout_err	When set to 1, it indicates that a CRC timeout error has occurred.
[0]	crc_chksum_err	When set to 1, it indicates that a CRC checksum error has occurred.

A stability error on the AXI5-Stream iritdest_*_ crc signal, results in bit[0] being set.

Page 2 reads for interrupt protection

If a GIC configuration does not enable interrupt protection for the block being accessed, FMU_SMRDATA interrupt protection page 2 returns zero and FMU_STATUS.PROTID_ERR returns `0b0`. The `spi_protection_type`, `rlt_spi_protection_type`, and `ppi_protection_type` parameters control whether a configuration supports interrupt protection.

Table 5-157: FMU_SMRDATA bit descriptions for page 2 (FMU_SMRD.PAGEID==2)

Bits	Name	Description
[31]	error_INTID_valid	Indicates whether an INTID-specific error has occurred: **0** Error valid bit is not set for the reported INTID. **1** Error valid bit is set for the reported INTID. See Interrupt error recovery on page 349 for more information about INTID-specific errors.

Bits	Name	Description
[30]	error_overflow	Returns the error overflow bit status: **0** No additional INTIDs have had an error, other than the error that error_INTID reports. **1** An error has occurred in one or more different INTIDs, including the error that error_INTID reports. Multiple errors on the same INTID do not set error_overflow, but those errors set the OFX, OFB, or OF bit in FMU_ERR<n>STATUS.
[29]	DetectionPaused	Detection is paused for one or more INTIDs.
[28]	-	Reserved

[27:16]	error_INTID	This value is the wire index for the protected interrupts on either a:
		***GIC Cluster Interface* (GCI)**
		For PPIs, this field returns an ID that indexes sequentially across all PPIs, for each PE that a GCI supports. Therefore, to calculate the PPI INTID:
		`index = error_INTID % GICR_CFGID1.PPIs_Per_Processor`
		`if index < 16 then`
		`  INTID = index + 16`
		`else`
		`  INTID = index + 1040`
		`end`
		To determine the PE that corresponds to this error, use:
		`CPUID = INT(error_INTID / GICR_CFGID1.PPIs_Per_Processor)`
		SPI Collator
		For SPIs, this field returns the interrupt ID on an SPI Collator that has an error. The sum of *error_INTID* and *INTID_base_offset* provides the INTID value.

[15:11]	-	Reserved
[10:0]	INTID_base_offset	Returns the base offset of the interrupt signal. For PPIs, this field returns zero. For SPIs, this field returns the base offset ID of the SPI Collator.

Error overflow can be set only when valid is set. Overflow means that a different INTID to the reported INTID also had an error.

If the FMU is reporting an interrupt protection error but error_INTID_valid reads back as 0, this means that the observed interrupt protection error is not associated with a specific INTID.

Page 3 reads for CPUIF protection

In the following table, `NUM_EXT_ERR_IF` is a build-time option of the CPUIF protection block. See 6.4 External error inputs on page 357 for more information.

Table 5-158: FMU_SMRDATA bit descriptions for page 3 (FMU_SMRD.PAGEID==3)

Bits	Name	Description
[NUM_EXT_ERR_IF + 23:24]	ext_err	When set to 1, it indicates that an external error has occurred on a CPUIF protection block.
[23]	lockstep_err	When set to 1, it indicates that a lock-step error has occurred.
[22]	cpu_parity_err	When set to 1, it indicates that a CPU parity error has occurred.

Bits	Name	Description
[21]	ci_parity_err	When set to 1, it indicates that a GCI parity error has occurred.
[20]	axit_crc_err	When set to 1, it indicates that an AXI5-Stream CRC error has occurred.
[19]	dft_err	When set to 1, it indicates that a DFT error has occurred.
[18]	qch_err	When set to 1, it indicates that a Q-Channel error has occurred.
[17]	clk_err	When set to 1, it indicates that a clock error has occurred.
[16]	reset_err	When set to 1, it indicates that a reset error has occurred.
[NUM_EXT_ERR_IF + 7:8]	ext_err_enabled	When set to 1, it indicates that an external interface error can generate a CPUIF protection fault.
[7]	lockstep_err_enabled	When set to 1, it indicates that a lock-step error can generate a CPUIF protection fault.
[6]	cpu_parity_err_enabled	When set to 1, it indicates that a CPU parity error can generate a CPUIF protection fault.
[5]	ci_parity_err_enabled	When set to 1, it indicates that a GCI parity error can generate a CPUIF protection fault.
[4]	axit_crc_err_enabled	When set to 1, it indicates that an AXI5-Stream CRC error can generate a CPUIF protection fault.
[3]	dft_err_enabled	When set to 1, it indicates that a DFT error can generate a CPUIF protection fault.
[2]	qch_err_enabled	When set to 1, it indicates that a Q-Channel error can generate a CPUIF protection fault.
[1]	clk_err_enabled	When set to 1, it indicates that a clock error can generate a CPUIF protection fault.
[0]	reset_err_enabled	When set to 1, it indicates that a reset error can generate a CPUIF protection fault.

Page 4 reads for CPUIF protection

The following table shows the page 4 reads for a CPUIF protection.

Table 5-159: FMU_SMRDATA bit descriptions for page 4 (FMU_SMRD.PAGEID==4)

Bits	Name	Description
[31]	deadlock_error_seen	When set to 1, it indicates the occurrence of a deadlock error.
[30]	deadlock_detection_enabled	When set to 1, it indicates that deadlock detection is active.
[29]	deadlock_correction_enabled	When set to 1, it indicates that deadlock correction is active.
[28:20]	-	Reserved
[19:5]	iritready_timeout	Returns the value of the iritready signal timeout. The timeout value is 8192 × (iritready_timeout + 1) − 1 cycles.
[4:0]	interbeat_timeout	Returns the value of the interbeat timeout. The timeout value is 256 × (interbeat_timeout + 1) − 1 cycles.

Accessibility

FMU_SMRDATA is accessible only by Secure accesses.

5.12.13 FMU_STATUS, FMU Status Register

This register monitors whether there are any outstanding AXI5-Stream messages waiting for responses.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-110: FMU_STATUS bit assignments

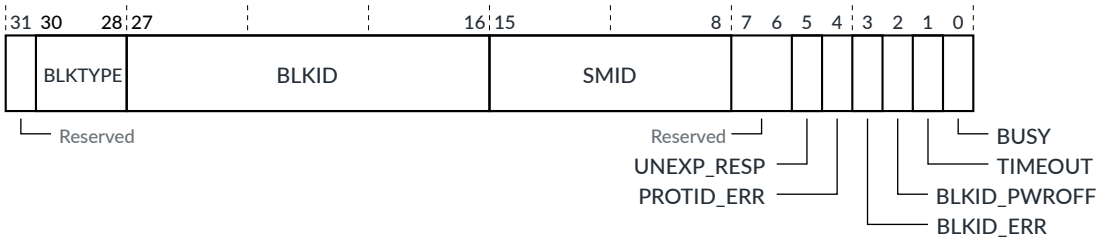


Table 5-160: FMU_STATUS bit descriptions

Bits	Name	Description
[31]	-	Reserved, RAZ
[30:28]	BLKTYPE	Block type identifier from the last response.
[27:16]	BLKID	Block identifier from the last response. However, during the ERRUPDATE mechanism, this field always indicates a valid block and has the following behavior: • If FMU_ERRUPDATE.MODE == 1 and the FMU receives a “block powered down” message, then this field contains the BLKID of the powered down block. • If the FMU receives an “invalid BLKID” message, then this field contains the last valid BLKID. • If the update process reaches the final BLKID and it receives a “Success” response, then this block is the last valid BLKID. The final BLKID is ($2^{BLKID\_WIDTH}-1$).
[15:8]	SMID	Protection mechanism identifier (PROTID) from the last response. However, if BLKID_ERR == 1 or BLKID_PWROFF == 1, then the SMID is not valid.
[7:6]	-	Reserved, RAZ

Bits	Name	Description
[5]	UNEXP_RESP	Indicates whether the last command received an unexpected response: **0** The previous command received a response that was expected. **1** The previous command failed because the FMU received an unexpected response, possibly due to a timeout.
[4]	PROTID_ERR	Indicates whether the last read or write command failed because the PROTID and PAGEID values were not valid: **0** The previous read or write command was sent using valid PROTID and PAGEID values. **1** The previous read or write command failed because the PROTID or PAGEID fields specified were set to an invalid value. For reads, this value also indicates that FMU_SMRDATA is invalid. Different protection mechanisms have a maximum PAGEID of 0, 1, or 2. Accessing a non-valid page causes the FMU to report an error.
[3]	BLKID_ERR	Indicates whether the last command failed because the destination block did not exist: **0** The previous command was sent to a valid destination block. **1** The previous command failed because the block that the BLKID field specified was set to an invalid value.
[2]	BLKID_PWROFF	Indicates whether the last command failed because the destination block was powered down: **0** The previous command was sent to a block that was powered up. **1** The previous command failed because the block that the BLKID field specified was powered down.
[1]	TIMEOUT	Indicates whether the last AXI5-Stream response timed out: **0** The FMU received the response for the previous AXI5-Stream message in ≤ FMU_TIMEOUT.DURATION clock cycles. **1** The FMU did not receive the response for the previous AXI5-Stream message in ≤ FMU_TIMEOUT.DURATION clock cycles. This value causes the FMU to set BUSY = 0.
[0]	BUSY	Indicates if the FMU is busy: **0** FMU is not busy. **1** FMU is busy processing the previous command.

Accessibility

FMU_STATUS is accessible only by Secure accesses.

5.12.14 FMU_KEY, FMU Key register

This register receives the unlock key that is required for writes to FMU registers to be successful. This register reads as 0 if the FMU register file is locked.

Software does not need to write to FMU_KEY when it performs FMU reads.

To handle 64-bit write accesses to 64-bit RAS registers, the key register is not affected by writes to the upper 32 bits of a 64-bit RAS register. This design functionality copes with the situation where an APB bridge reverses the order of the two 32-bit writes in a 64-bit write access.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-111: FMU_KEY bit assignments

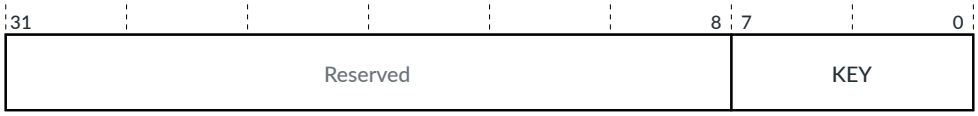


Table 5-161: FMU_KEY bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ
[7:0]	KEY	Writing the correct key to this field enables the next write to any other writable FMU register to succeed. The register file is unlocked when a write to FMU_KEY occurs that satisfies all the following conditions: • Is Secure. • Is for 32 bits. • Bits[7:0] are `0xBE`. If the register file is unlocked, the FMU_KEY register reads as `0x000000BE`. Otherwise, the FMU_KEY register reads as `0x00000000`. The FMU_KEY register automatically locks after any Secure write access with correct strobes, even if the write is ignored. For example, the write might be ignored if it accesses an invalid address. The only exception is writing to the upper 32 bits of the 64-bit RAS registers, FMU_ERR<n>CTLR and FMU_ERR<n>STATUS. See 6.1.8 Lock and key mechanism on page 353.

Accessibility

FMU_KEY is accessible only by Secure accesses.

5.12.15 FMU_TIMEOUT, Timeout duration register

When FMU_STATUS.BUSY == 1, this register controls the duration before the FMU sets FMU_STATUS.TIMEOUT = 1.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

After a write to this register, poll FMU_STATUS.BUSY to ensure that the effect of the write is complete.

Bit description

Figure 5-112: FMU_TIMEOUT bit assignments

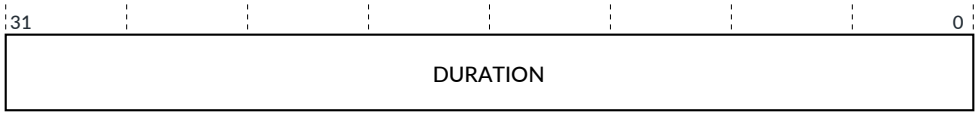


Table 5-162: FMU_TIMEOUT bit descriptions

Bits	Name	Description
[31:0]	DURATION	Timeout count duration in FMU clock cycles. If FMU_STATUS.BUSY asserts for longer than the value of this field, then an AXI5-Stream timeout occurs and the FMU sets: • FMU_STATUS.BUSY = 0 • FMU_STATUS.TIMEOUT = 1 When TIMEOUT == 1, the FMU stops waiting for the AXI5-Stream response, so software can then start a new AXI5-Stream request. The initial value on reset is `0xFFFFFFFF`, which provides the longest possible timeout allowed.

Accessibility

FMU_TIMEOUT is accessible only by Secure accesses.

5.12.16 FMU_ERRUPDATE, Error Update register

This register updates an error record pair FMU_ERR<n>STATUS and FMU_ERR<n+1>STATUS, with all the reported error states. If software clears the FMU_ERR<n>STATUS.OFX bit, then it can use FMU_ERRUPDATE to discover the source of the error that caused the OFX to resend its error.

The update process sequentially sends an FMU_CTRL_ACCESS message that contains PROTID=255, to all fault collators on the error record pair. On receiving this message, the fault collator sends any pending errors, up to a maximum of two critical errors and two non-critical errors. When the FMU receives the errors it updates FMU_ERR<n>STATUS and FMU_ERR<n +1>STATUS.

When the error update process starts, the FMU sets FMU_STATUS.BUSY=1, and the bit remains set until completion. The FMU_STATUS.BLKID indicates the progress of the update process. After the process completes, if:

- MODE==0, the FMU_STATUS.BLKID shows the last valid BLKID.
- MODE==1, the FMU_STATUS.BLKID shows the BLKID of the first powered-down block.

The error update process stops when:

- It receives a response packet with “BLKID invalid”.
- MODE=1, and it receives a response packet with “powered down block”.
- BLKTYPE is internal FMU, and it checks the only valid BLKID (0).
- It reaches the last possible BLKID based on the `BLKID_WIDTH` parameter.
- FMU_STATUS.BUSY==1 for a duration that causes a timeout to occur.

During the error update process, the FMU does not record any “powered down” or “invalid block” responses as errors, because the mechanism requires those responses. However, if the first BLKID receives a “powered down” or “invalid block” response, then the FMU reports that error. The *first BLKID* is the value that software wrote.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-113: FMU_ERRUPDATE bit assignments

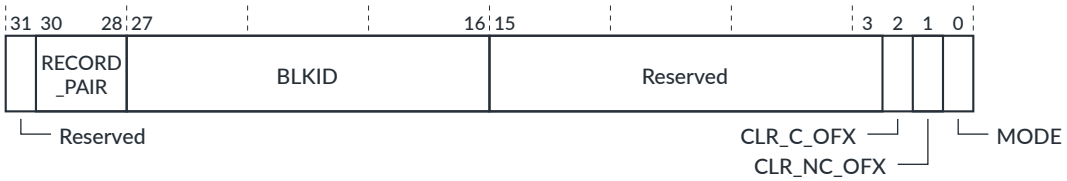


Table 5-163: FMU_ERRUPDATE bit descriptions

Bits	Name	Description
[31]	-	Reserved, RAZ
[30:28]	RECORD_PAIR	This field contains the ID of the error record pair to update: **0** GICD **1** Wake Request **2** SPI Collator **3** *GIC Cluster Interface* (GCI) **4** ITS **5** FMU **6, 7** Reserved The updated error records are n and n+1, where n = 2 × RECORD_PAIR.
[27:16]	BLKID	This field sets the starting block identifier for the error record pair update process.
[15:3]	-	Reserved, RAZ
[2]	CLR_C_OFX	Clears the FMU_ERR<n>STATUS.OFX bit of the critical error record (n = 2 × RECORD_PAIR).
[1]	CLR_NC_OFX	Clears the FMU_ERR<n>STATUS.OFX bit of the non-critical error record (n = 2 × RECORD_PAIR + 1).
[0]	MODE	Controls whether the BLKID incrementation stops due to an invalid BLKID or a powered down block: **0** Increment the BLKID value until either BLKID wraps or an invalid BLKID occurs, that is, FMU_STATUS.BLKID_ERR == 1. **1** Increment the BLKID value until either BLKID wraps or the process encounters a powered down block, that is, FMU_STATUS.BLKID_PWROFF == 1.

Accessibility

FMU_ERRUPDATE is accessible only by Secure accesses.

5.12.17 FMU_FCTLR, Function Control Register

This register controls clock gating of the FMU.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit

Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-114: FMU_FCTLR bit assignments

Table 5-164: FMU_FCTLR bit assignments

Bits	Name	Description
[31:1]	-	Reserved, RES0
[0]	CGO	The FMU clock gate override: **0** Use full clock gating. **1** Leave clock running. If clock gates are not implemented, then you must use this value.

Accessibility

FMU_FCTLR is accessible only by Secure accesses.

5.12.18 FMU_ERRDEVID, Device configuration register

This register returns the number of error records in the FMU.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-115: FMU_ERRDEVID bit assignments

Table 5-165: FMU_ERRDEVID bit descriptions

Bits	Name	Description
[31:16]	-	Reserved, RAZ
[15:0]	NUM	This field returns `0x000C` because the FMU has 12 error records. The FMU_ERRGSR effectively lists the error records, 0-11, and the block a record associates with.

Accessibility

FMU_ERRDEVID is accessible only by Secure accesses.

5.12.19 FMU_PIDR2, Peripheral ID2 Register

This register returns byte[2] of the peripheral ID. The FMU_PIDR2 register is part of the peripheral identification registers.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.12 FMU register summary on page 291 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-116: FMU_PIDR2 bit assignments

Table 5-166: FMU_PIDR2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ

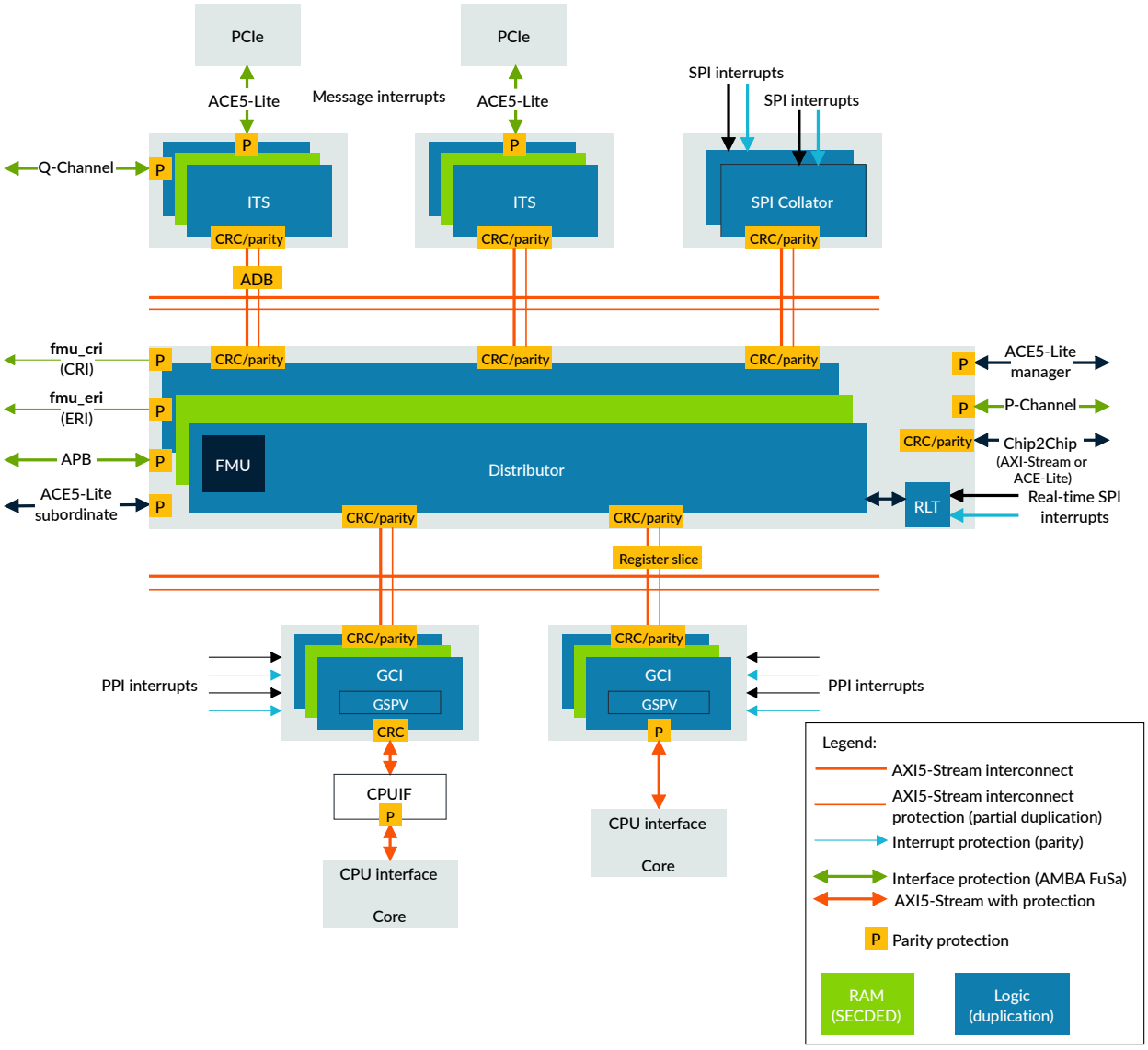
Bits	Name	Description
[7:4]	REVISION	Identifies the major revision of the FMU: **0x0** r0
[3]	JEDEC	Indicates that a JEDEC-assigned JEP106 identity code is used.
[2:0]	DES_1	Bits[6:4] of the JEP106 identity code. Bits[3:0] of the JEP106 identity code are assigned to FMU_PIDR1[7:4].

6. Functional safety in GIC-720AE

GIC-720AE is a version of GIC-700 with FuSa detection features added. All FuSa features are "bolted on" to GIC-700 and do not alter the original GIC-700 functionality.

The following figure shows where the main protection mechanisms of GIC-720AE reside.

Figure 6-1: Protection mechanism distribution



GIC-720AE contains the following FuSa protection mechanisms.

Lock-step logic protection

The logic is protected with duplicated logic running in lock-step with a temporal delay.

RAM protection

The RAMs are shared between the lock-stepped primary and secondary blocks and are protected with SECDED ECC. The address is further protected with address parity.

AMBA® AXI5-Stream interconnect protection

The AXI5-Stream interconnect that connects the GIC blocks, is protected with either:

- AMBA parity for simple point-to-point connections
- End-to-end CRC for switched connections or when:
  - ADB domain bridges are required.
  - `structure == wrap` and the GIC is configured to reduce the number of AXI5-Stream interfaces on the GCI.

AMBA external interface protection

All external AMBA interfaces are protected with AMBA parity signals. AMBA parity protects point-to-point connections consisting of wires and buffers only, and no gates. This protection includes the ACE5-Lite, AXI5-Stream, Q-Channel, P-Channel, *Cross-Chip* (CC), and APB external ports.


Note

- The P-Channel protection is for cross-chip functions on the Distributor.
- Figure 6-1: Protection mechanism distribution on page 331 shows Q-Channel parity protection that is enabled on only one ITS block. However, when `qch_protection_type == 1`, the Q-Channel protection is present for all Q-Channel interfaces.

AMBA cross-chip interface protection

End-to-end CRC protection for cross-chip connections that use either ACE5-Lite or AXI5-Stream.

PPI and SPI source interrupt parity protection

The PPI and SPI interrupt input sources are protected with optional parity protection. There is one parity bit for each PPI and SPI input pin.

CPUIF protection

CPUIF protection provides protection for the AXI5-Stream interface between a GCI and a processor. CPUIF protection is required when there is not a point-to-point connection between the GCI and the processor, and it allows CRC protection to be used on the AXI5-Stream interface.

Behavioral separation

Multi view support enables behavioral separation between interrupts and PEs in different views.

GSPV protection

The *GIC Stream Protocol Validator* (GSPV) gives protocol level error detection and correction for the GIC Stream protocol, including the lower level AXI5-Stream protocol rules. Each *GIC Cluster Interface* (GCI) contains a GSPV.

This protection supports mixed criticality systems, where the processor has a lower ASIL level than ASIL D.

AXI5-Stream PING/ACK

GIC-720AE contains a watchdog-based PING/ACK mechanism. This mechanism protects against systematic errors on the interconnect that connects the various GIC blocks. If the mechanism does not receive a response within the programmable timeout window, it reports a fault.

The AXI5-Stream protection contains a PING/ACK mechanism, as part of CRC end-to-end protection, where separate CRC packets are sent to check the data integrity of data packets and protect against spurious packets and packet loss.

Clocks and resets

The clocks and resets are duplicated. The internally gated clocks operate with a temporal delay of two. That is, the secondary logic operates two cycles later than the primary logic.

Fault Management Unit

The *Fault Management Unit* (FMU) resides in the Distributor. It processes faults that the protection mechanisms detect from all GIC blocks. The FMU records the fault syndrome in the error records and reports the fault using *Error Recovery Interrupt* (ERI) and *Critical Error Interrupt* (CRI). There are also FMU registers that enable fault injection and clearing for each protection mechanism. The FMU communicates with an external processor, which is responsible for handling errors after fault detection, through an APB port. The APB port is for FuSa purposes and does not exist on the GIC-700, the non-FuSa version.

Protection mechanisms

For a detailed list of the protection mechanisms available in GIC-720AE, see the *Fault Detection and Control mechanisms* chapter in the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Safety Manual*.

6.1 Fault Management Unit

The FMU is part of the GIC Distributor (GICD) component.

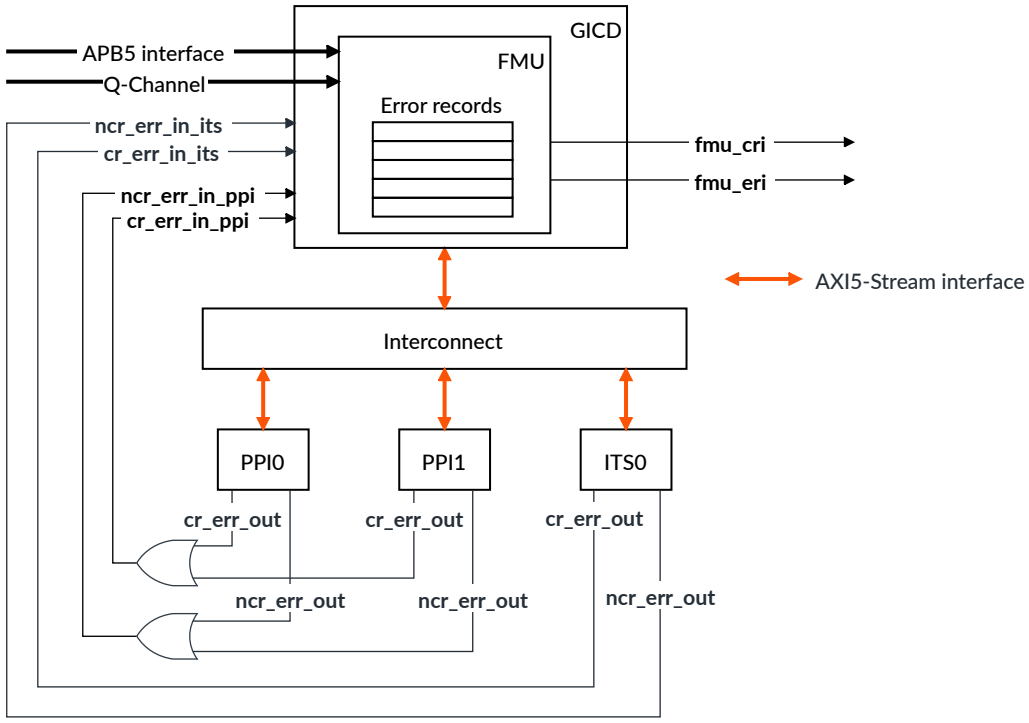
The FMU implements the following functionality in GIC-720AE:

- Dedicated APB5 interface to access error records and other registers.
- Routes all errors to the Safety Island, if enabled.
- Provides software the means to enable or disable a protection mechanism within a GIC block.
- Receives error signaling from all protection mechanisms within other GIC blocks.
- Maintains error records for each GIC block type, for software inspection and provides information on the source of the error.
- Retains error records across functional reset.

- Enables software error recovery testing by providing error injection capabilities in a protection mechanism.

The following figure shows the FMU and its interconnections.

Figure 6-2: FMU interconnections



6.1.1 FMU APB5 interface

The programmer view registers inside the FMU are accessible through an APB5 interface that is protected with AMBA parity extensions.

The APB5 completer interface width is 32 bits. Some of the FMU registers are 64 bits wide, so two 32-bit APB accesses, in any order, are necessary for reads or writes of those registers.

The APB5 port allows only Secure access to the FMU. To implement this access restriction, the pprot[1] signal is checked during an access. If the access fails the security check, the GIC does not use the pslverr signal to indicate this error condition, the pslverr signal remains LOW.

6.1.2 FMU Q-Channel

The *Fault Management Unit* (FMU) has a Q-Channel interface which controls requests from an external clock gating source.

If the FMU is busy, the Q-Channel interface asserts the qdeny signal to deny an external request to gate its clock. When an external request occurs, the interface requests a wakeup by asserting the qactive signal.

The qreqn input signal is synchronized to the FMU.

6.1.3 Error signaling

This section describes how GIC blocks signal errors, and how the FMU reports these errors.

Error signaling from a GIC block to the FMU

GIC-720AE implements several protection mechanisms in each GIC block to protect against random transient or permanent errors. Each protection mechanism sends an error signal to the fault collator in its GIC block. The GIC block then forwards the error signal to the FMU in the central GIC Distributor using the existing AXI5-Stream interface.

In addition to reporting errors through the AXI5-Stream interconnect, each remote GIC block has a cr_err_out and ncr_err_out output signal that indicates either a critical or non-critical error within its block.

As the GIC exits reset, it sets all protection mechanisms to report a critical error, except for the RAM SEC protection mechanisms that report a non-critical error. Software can use the 5.12.8 FMU_SMCR, Safety Mechanism Set Criticality Register on page 306 to alter these criticality assignments.

The cr_err_out and ncr_err_out signals must be ORed for all blocks of that block type, and routed back to the cr_err_in_* and ncr_err_in_* signals on the GICD. The remote GIC block keeps the cr_err_out or ncr_err_out signal asserted until the error recovery software clears the error.

Error signaling by the FMU

When a protection mechanism detects an error, it forwards the error to the FMU. If the FMU is enabled, it signals the error to the entire system using the error interrupt signals. These signals are:

- Error recovery interrupt, fmu_eri signal (ERI)
- Critical error interrupt, fmu_cri signal (CRI)

The ERI and CRI interrupts are enabled on reset. Software can use FMU_ERR<n>CTLR to disable the error reporting through ERI or CRI.

Non-critical errors are reported using ERI and critical errors are reported using CRI. The FMU_ERR<n>CTLR.CI and FMU_ERR<n>CTLR.UI bits control this reporting. The grouping of the errors into these two categories can be helpful in redirecting these errors to different error

recovery handlers, either based on the criticality of the errors or other factors that are known at the system level.

6.1.4 Error record format

The FMU contains one error record for each GIC block type.

GIC-720AE faults are recorded in error records.

The error record registers are accessible through the APB5 interface on the GICD. The FMU has a different reset input signal than the GIC, so that the error record retains its state even when the GIC block is being reset.

The following table lists the error record IDs for the different GIC block types. Some GIC configurations might not contain all of these block types.

Table 6-1: Error record block type IDs

FMU_ERRUPDATE.RECORD_PAIR	Block type	Error record ID, <n>	Error criticality
0	GICD	0	Critical
		1	Non-critical
1	Wake Request	2	Critical
		3	Non-critical
2	SPI Collator	4	Critical
		5	Non-critical
3	GCI	6	Critical
		7	Non-critical
4	ITS	8	Critical
		9	Non-critical
5	FMU	10	Critical
		11	Non-critical

6.1.5 FMU reset

When the FMU reports multiple uncorrectable errors, the error recovery procedure might require the GIC to be reset. To facilitate this situation, the FMU has a separate reset input signal, fmu_reset_n.

This reset differs from the GIC functional reset, reset_n signal. It allows the FMU to retain error records across GIC functional reset.

6.1.6 Protection mechanism IDs

The GIC assigns an ID for each protection mechanism in a functional block. For each protection mechanism ID we provide a description and the recommended recovery process.

GIC Cluster Interface (GCI) protection mechanisms

The following table lists the IDs for each protection mechanism of a GCI.

Table 6-2: GCI protection mechanisms

ID	Protection name	Description	Recovery
0	Invalid protection ID	Reserved value	-
1	SM_CLOCK_CI	Clock error	Block reset on page 347
2	SM_RESET_CI	Reset error	Reset error recovery on page 350
3	SM_LOCKSTEP_CI	Lock-step error	Block reset on page 347
4	SM_AXITPAR_CI_ICPD	Interface to GICD AXI5-Stream parity error.	Block reset on page 347
5	SM_AXITPAR_CI_IRI	Interface to CPU AXI5-Stream parity error.	Block reset on page 347
6	SM_AXITCRC_CI_ICPD	Interface to GICD AXI5-Stream CRC error.	CRC error recovery on page 348
7	SM_AXITCRC_CI_IRI	Interface to CPU AXI5-Stream CRC error.	CRC error recovery on page 348
8	SM_QCH_CI	GCI clock Q-Channel	Q-Channel error recovery on page 350
9	SM_INT_CI_PPI	Interrupt error	Interrupt error recovery on page 349
10	SM_CPU_ACTIVE_CI	Duplication error on asynchronous input.	Asynchronous input error recovery on page 347
11	SM_DFT_CI	DFT error	Block reset on page 347
12	SM_MBIST_CI	MBIST error	Block reset on page 347

ID	Protection name	Description	Recovery
13	SM_SECD_CI	GCI RAM single error correction in data bit.	RAM error recovery on page 350
14	SM_SEDA_CI	GCI RAM single error detection in address bit.	RAM error recovery on page 350
15	SM_DED_CI	GCI RAM double error detection.	RAM error recovery on page 350
16	SM_GSPV_AXIT	GIC Stream Protocol Validator AXI5-Stream error.	GSPV error recovery on page 349
17	SM_GSPV_PROTO	GIC Stream Protocol Validator protocol error.	GSPV error recovery on page 349
18	SM_EXT0_CI	Auxiliary external error 0	External error recovery on page 348
19	SM_EXT1_CI	Auxiliary external error 1	External error recovery on page 348
20	SM_LPD_CI	LPD error	LPD error recovery on page 349
21	SM_LPD_CI_IC	LPD error GIC internal interconnect	LPD error recovery on page 349
22-37	SM_CPUIF_PROT	Up to 16 *CPU interface* (CPUIF) protection blocks. The number of CPUIF protection blocks, or valid SM_CPUIF_PROT IDs, depends on the number of buses that are configured in the GCI. There can be up to 16 CPUIF protection blocks on one GCI. When a CPUIF reports an error through its SM_CPUIF_PROT, that SM_CPUIF_PROTID does not report any more errors until the source of the error has been cleared. Software can either clear or disable the error, with a write to FMU_SMWDATA in page-3, for that SM_CPUIF_PROT.	CPUIF error recovery on page 348 of connected GCI
255	-	Software can use this setting to: • Enable or disable the error output signals on this GCI. See 6.1.6.4 Enabling or disabling both error signals on a block on page 351. • Resend any outstanding errors on this GCI. See 6.1.6.5 Discovering the active errors on a block on page 352.	-

ITS protection mechanisms

The following table lists the IDs for each protection mechanism of an ITS.

Table 6-3: ITS protection mechanisms

ID	Protection name	Description	Recovery
0	Invalid protection ID	Reserved value	-

ID	Protection name	Description	Recovery
1	SM_CLOCK_ITS	Clock error	Block reset on page 347
2	SM_RESET_ITS	Reset error	Reset error recovery on page 350
3	SM_LOCKSTEP_ITS	Lock-step error	Block reset on page 347
4	SM_ACELMPAR_ITS	ACE5-Lite manager parity error. This protection mechanism is present in bypass configurations only.	Block reset on page 347
5	SM_ACELSPAR_ITS	ACE5-Lite subordinate TranslateR parity error	Block reset on page 347
6	SM_AXITPAR_ITS_ICID	ITS to GICD AXI5-Stream parity error.	Block reset on page 347
7	SM_AXITPAR_ITS_MSIR	Direct port AXI5-Stream parity error.	Block reset on page 347
8	SM_AXITCRC_ITS_ICID	ITS to GICD AXI5-Stream CRC error.	CRC error recovery on page 348
9	SM_QCH_ITS_CLK	Clock Q-Channel	Q-Channel error recovery on page 350
10	SM_DFT_ITS	DFT interface error	Block reset on page 347
11	SM_MBIST_ITS	MBIST interface error	Block reset on page 347
12	SM_SECD_ITS_DID	*Device ID* (DID) RAM single error correction in data bit.	RAM error recovery on page 350
13	SM_SEDA_ITS_DID	DID RAM single error detection in address bit.	RAM error recovery on page 350
14	SM_DED_ITS_DID	DID RAM double error detection	RAM error recovery on page 350
15	SM_SECD_ITS_VID	*Event ID* (VID) RAM single error correction in data bit.	RAM error recovery on page 350
16	SM_SEDA_ITS_VID	VID RAM single error detection in address bit.	RAM error recovery on page 350
17	SM_DED_ITS_VID	VID RAM double error detection	RAM error recovery on page 350
18	SM_SECD_ITS_COL	*Collection* (COL) RAM single error correction in data bit.	RAM error recovery on page 350
19	SM_SEDA_ITS_COL	COL RAM single error detection in address bit.	RAM error recovery on page 350
20	SM_DED_ITS_COL	COL RAM double error detection	RAM error recovery on page 350
21	SM_EXT0_ITS	Auxiliary external error 0	External error recovery on page 348
22	SM_EXT1_ITS	Auxiliary external error 1	External error recovery on page 348
23	SM_LPD_ITS	LPD error	LPD error recovery on page 349

ID	Protection name	Description	Recovery
24	SM_LPD_ITS_IC	LPD error GIC internal interconnect	LPD error recovery on page 349
255	-	Software can use this setting to: • Enable or disable the error output signals on this ITS. See 6.1.6.4 Enabling or disabling both error signals on a block on page 351. • Resend any outstanding errors on this ITS. See 6.1.6.5 Discovering the active errors on a block on page 352.	-

GICD protection mechanisms

The following table lists the IDs for each protection mechanism of the GICD.

Table 6-4: GICD protection mechanisms

ID	Protection name	Description	Recovery
0	Invalid protection ID	Reserved value	-
1	SM_CLOCK_GICD	Clock error	Block reset on page 347
2	SM_RESET_GICD	Reset error	Reset error recovery on page 350
3	SM_LOCKSTEP_GICD	Lock-step error	Block reset on page 347
4	SM_ACELMPAR_GICD	ACE5-Lite manager parity error	Block reset on page 347
5	SM_ACELSPAR_GICD	ACE5-Lite subordinate parity error	Block reset on page 347
6	SM_ACELSPAR_CC	ACE5-Lite subordinate cross-chip parity error	Block reset on page 347
7	SM_AXITPAR_GICD_ICDW	Wake Request AXI5-Stream parity error	Block reset on page 347
8	SM_AXITPAR_GICD_ICDC	SPI Collator AXI5-Stream parity error	Block reset on page 347
9	SM_AXITPAR_GICD_ICDP	PPI AXI5-Stream parity error	Block reset on page 347
10	SM_AXITPAR_GICD_ICDR	Cross-chip AXI5-Stream parity error. This protection mechanism is present in cross-chip AXI5-Stream configurations only.	Block reset on page 347
11	SM_AXITPAR_GICD_ICDI	ITS AXI5-Stream parity error	Block reset on page 347
12	SM_AXITCRC_GICD_ICDW	Wake Request AXI5-Stream CRC error	CRC error recovery on page 348
13	SM_AXITCRC_GICD_ICDC	SPI Collator AXI5-Stream CRC error	CRC error recovery on page 348
14	SM_AXITCRC_GICD_ICDP	PPI AXI5-Stream CRC error	CRC error recovery on page 348
15	SM_AXITCRC_GICD_ICDR	Cross-chip CRC error. This protection mechanism is present in all cross-chip configurations, that is AXI5-Stream or ACE5-Lite.	CRC error recovery on page 348

ID	Protection name	Description	Recovery
16	SM_AXITCRC_GICD_ICDI	ITS AXI5-Stream CRC error	CRC error recovery on page 348
17	SM_PCH_GICD	GICD power P-Channel error	P-Channel recovery on page 349
18	SM_QCH_GICD_CLK	GICD clock Q-Channel error	Q-Channel error recovery on page 350
19	SM_QCH_GICD_ITS	ITS Q-Channel error	Q-Channel error recovery on page 350
20	SM_INT_GICD_RLTSPI	Real-time SPI interrupt wire error	Interrupt error recovery on page 349
21	SM_DUPIN_SAMPLEREQ	Sample request input error	Asynchronous input error recovery on page 347
22	SM_DFT_GICD	DFT interface error	Block reset on page 347
23	SM_MBIST_GICD	MBIST interface error	Block reset on page 347
24	SM_SECD_GICD_SPI0	SPI0 RAM single error correction in data bit.	RAM error recovery on page 350
25	SM_SEDA_GICD_SPI0	SPI0 RAM single error detection in address bit.	RAM error recovery on page 350
26	SM_DED_GICD_SPI0	SPI0 RAM double error detection	RAM error recovery on page 350
27	SM_SECD_GICD_SPI1	SPI1 RAM single error correction in data bit.	RAM error recovery on page 350
28	SM_SEDA_GICD_SPI1	SPI1 RAM single error detection in address bit.	RAM error recovery on page 350
29	SM_DED_GICD_SPI1	SPI1 RAM double error detection	RAM error recovery on page 350
30	SM_SECD_GICD_LPI0	LPI0 RAM single error correction in data bit.	RAM error recovery on page 350
31	SM_SEDA_GICD_LPI0	LPI0 RAM single error detection in address bit.	RAM error recovery on page 350
32	SM_DED_GICD_LPI0	LPI0 RAM double error detection	RAM error recovery on page 350
33	SM_SECD_GICD_LPI1	LPI1 RAM single error correction in data bit.	RAM error recovery on page 350
34	SM_SEDA_GICD_LPI1	LPI1 RAM single error detection in address bit.	RAM error recovery on page 350
35	SM_DED_GICD_LPI1	LPI1 RAM double error detection	RAM error recovery on page 350
36	SM_SECD_GICD_LPI2	LPI2 RAM single error correction in data bit.	RAM error recovery on page 350
37	SM_SEDA_GICD_LPI2	LPI2 RAM single error detection in address bit.	RAM error recovery on page 350

ID	Protection name	Description	Recovery
38	SM_DED_GICD_LPI2	LPI2 RAM double error detection	RAM error recovery on page 350
39	SM_SECD_GICD_LPI3	LPI3 RAM single error correction in data bit.	RAM error recovery on page 350
40	SM_SEDA_GICD_LPI3	LPI3 RAM single error detection in address bit.	RAM error recovery on page 350
41	SM_DED_GICD_LPI3	LPI3 RAM double error detection	RAM error recovery on page 350
42	SM_SECD_GICD_TGT_LPI	TGT_LPI RAM single error correction in data bit.	RAM error recovery on page 350
43	SM_SEDA_GICD_TGT_LPI	TGT_LPI RAM single error detection in address bit.	RAM error recovery on page 350
44	SM_DED_GICD_TGT_LPI	TGT_LPI RAM double error detection	RAM error recovery on page 350
45	SM_SECD_GICD_TGT_SPI	TGT_SPI RAM single error correction in data bit.	RAM error recovery on page 350
46	SM_SEDA_GICD_TGT_SPI	TGT_SPI RAM single error detection in address bit.	RAM error recovery on page 350
47	SM_DED_GICD_TGT_SPI	TGT_SPI RAM double error detection	RAM error recovery on page 350
48	SM_SECD_GICD_SGI	SGI RAM single error correction in data bit.	RAM error recovery on page 350
49	SM_SEDA_GICD_SGI	SGI RAM single error detection in address bit.	RAM error recovery on page 350
50	SM_DED_GICD_SGI	SGI RAM double error detection	RAM error recovery on page 350
51	SM_SECD_GICD_VTGTVRES	*Virtual Target Residency* (VTGT_VRES) RAM single error correction in data bit.	RAM error recovery on page 350
52	SM_SEDA_GICD_VTGTVRES	VTGT_VRES RAM single error detection in address bit.	RAM error recovery on page 350
53	SM_DED_GICD_VTGTVRES	VTGT_VRES RAM double error detection	RAM error recovery on page 350
54	SM_SECD_GICD_VTGTVSTR	*Virtual Target Store* (VTGT_VSTR) RAM single error correction in data bit.	RAM error recovery on page 350
55	SM_SEDA_GICD_VTGTVSTR	VTGT_VSTR RAM single error detection in address bit.	RAM error recovery on page 350
56	SM_DED_GICD_VTGTVSTR	VTGT_VSTR RAM double error detection	RAM error recovery on page 350
57	SM_SECD_GICD_VTGT	*Virtual Target Search* (VTGT_SRCH) RAM single error correction in data bit.	RAM error recovery on page 350
58	SM_SEDA_GICD_VTGT	VTGT_SRCH RAM single error detection in address bit.	RAM error recovery on page 350
59	SM_DED_GICD_VTGT	VTGT_SRCH RAM double error detection	RAM error recovery on page 350
60	SM_SECD_GICD_PTS	*Pending Table System* (PTS) RAM single error correction in data bit.	RAM error recovery on page 350

ID	Protection name	Description	Recovery
61	SM_SEDA_GICD_PTS	PTS RAM single error detection in address bit.	RAM error recovery on page 350
62	SM_DED_GICD_PTS	PTS RAM double error detection	RAM error recovery on page 350
63	SM_SECD_GICD_VICM	*Virtual ITS Communication Module* (VICM) RAM single error correction in data bit.	RAM error recovery on page 350
64	SM_SEDA_GICD_VICM	VICM RAM single error detection in address bit.	RAM error recovery on page 350
65	SM_DED_GICD_VICM	VICM RAM double error detection	RAM error recovery on page 350
66	SM_SECD_GICD_VSPA	*Virtual SGI Pending Array* (VSPA) RAM single error correction in data bit.	RAM error recovery on page 350
67	SM_SEDA_GICD_VSPA	VSPA RAM single error detection in address bit.	RAM error recovery on page 350
68	SM_DED_GICD_VSPA	VSPA RAM double error detection	RAM error recovery on page 350
69	SM_SECD_GICD_CC	Cross-chip RAM single error correction in data bit.	RAM error recovery on page 350
70	SM_SEDA_GICD_CC	Cross-chip RAM single error detection in address bit.	RAM error recovery on page 350
71	SM_DED_GICD_CC	Cross-chip RAM double error detection	RAM error recovery on page 350
72	SM_INT_GICD_RLT	Real-time SPI interrupt processing error	Block reset on page 347
73	SM_EXT0_GICD	External error 0	External error recovery on page 348
74	SM_EXT1_GICD	External error 1	External error recovery on page 348
75	SM_LPD_GICD	LPD error	LPD error recovery on page 349
76	SM_LPD_GICD_IC	LPD error GIC internal interconnect	LPD error recovery on page 349
255	-	Software can use this setting to: • Enable or disable the error output signals on the GICD. See 6.1.6.4 Enabling or disabling both error signals on a block on page 351. • Resend any outstanding errors on the GICD. See 6.1.6.5 Discovering the active errors on a block on page 352.	-

SPI Collator protection mechanisms

The following table lists the IDs for each protection mechanism of an SPI Collator.

Table 6-5: SPI Collator protection mechanisms

ID	Protection name	Description	Recovery
0	Invalid protection ID	Reserved value	-
1	SM_CLOCK_SPIC	Clock error	Block reset on page 347

ID	Protection name	Description	Recovery
2	SM_RESET_SPIC	Reset error	Reset error recovery on page 350
3	SM_LOCKSTEP_SPIC	Lock-step error	Block reset on page 347
4	SM_AXITPAR_SPIC	Interface to GICD AXI5-Stream parity error.	Block reset on page 347
5	SM_AXITCRC_SPIC	Interface to GICD AXI5-Stream CRC error.	CRC error recovery on page 348
6	SM_QCH_SPIC_COL_CLK	Clock Q-Channel error	Q-Channel error recovery on page 350
7	SM_QCH_SPIC_COL	Power Q-Channel error	Q-Channel error recovery on page 350
8	SM_SPI_SPIC	Interrupt error	Interrupt error recovery on page 349
9	SM_EXT0_SPIC	External error 0	External error recovery on page 348
10	SM_EXT1_SPIC	External error 1	External error recovery on page 348
11	SM_LPD_SPIC	LPD error	LPD error recovery on page 349
12	SM_LPD_SPIC_IC	LPD error GIC internal interconnect	LPD error recovery on page 349
255	-	Software can use this setting to: • Enable or disable the error output signals on this SPI Collator. See 6.1.6.4 Enabling or disabling both error signals on a block on page 351. • Resend any outstanding errors on this SPI Collator. See 6.1.6.5 Discovering the active errors on a block on page 352.	-

Wake Request protection mechanisms

The following table lists the IDs for each protection mechanism of the Wake Request block.

Table 6-6: Wake Request protection mechanisms

ID	Protection name	Description	Recovery
0	Invalid protection ID	Reserved value	-
1	SM_CLOCK_WAKE	Clock error	Block reset on page 347
2	SM_RESET_WAKE	Reset error	Reset error recovery on page 350
3	SM_LOCKSTEP_WAKE	Lock-step error	Block reset on page 347
4	SM_AXITPAR_WAKE_ICWD	Interface to GICD AXI5-Stream parity error.	Block reset on page 347
5	SM_AXITCRC_WAKE_ICWD	Interface to GICD AXI5-Stream CRC error.	CRC error recovery on page 348
6	SM_QCH_WAKE	Q-Channel error	Q-Channel error recovery on page 350
7	SM_EXT0_WAKE	External error 0	External error recovery on page 348
8	SM_EXT1_WAKE	External error 1	External error recovery on page 348
9	SM_LPD_WAKE	LPD error	LPD error recovery on page 349
10	SM_LPD_WAKE_IC	LPD error GIC internal interconnect	LPD error recovery on page 349

ID	Protection name	Description	Recovery
255	-	Software can use this setting to: • Enable or disable the error output signals on the Wake Request. See 6.1.6.4 Enabling or disabling both error signals on a block on page 351. • Resend any outstanding errors on the Wake Request. See 6.1.6.5 Discovering the active errors on a block on page 352.	-

FMU protection mechanisms

The following table lists the IDs for each protection mechanism of the FMU.

Table 6-7: FMU protection mechanisms

ID	Protection name	Description	Recovery
0	Invalid protection ID	Reserved value	-
1	SM_CLOCK_FMU	Clock error	FMU reset on page 348
2	SM_RESET_FMU	Reset error	Reset error recovery on page 350
3	SM_LOCKSTEP_FMU	Lock-step error	FMU reset on page 348
4	SM_QCH_FMU	Q-Channel error	Q-Channel error recovery on page 350
5	SM_APBPTY_FMU	AMBA parity error	FMU APB recovery on page 348
6	SM_DFT_FMU	DFT protection error	FMU reset on page 348
7	SM_BRIDGEFMU_FMU	A consistency error on the FMU-side of the GIC-FMU bridge.	Full reset of GIC (not FMU)
8	SM_KEY_FMU	APB register write was prevented by the FMU_KEY register.	FMU APB access error recovery on page 348
9	SM_SECURITY_FMU	Non-Secure read or write access to a Secure FMU register.	FMU APB access error recovery on page 348
10	SM_APB_ACCESS_FMU	APB write error to an FMU register. This error occurs when any of the following is true: • A write to an invalid address. An invalid address is defined as a register that is not shown in Table 5-131: FMU register summary on page 292. • A write to a read-only FMU register. Read accesses cannot generate this error. Therefore, for discovery purposes, software can always read any FMU RAS registers.	FMU APB access error recovery on page 348

ID	Protection name	Description	Recovery
11	SM_APB_FIELD_FMU	APB field invalid. This error occurs when any of the following is true: • BLKTYPE field value selects a block type that the GIC configuration does not support. • BLKTYPE > 5 • PAGEID field value > 15 • BLKID bits exceed the `BLKID_WIDTH` parameter. `BLKID_WIDTH` is the number of bits required to store the maximum BLKID. • BLKTYPE == 0 (GICD) and BLKID != 0 • BLKTYPE == 1 (Wake Request) and BLKID != 0 • BLKTYPE == 5 (FMU) and BLKID != 0 • PROTID == 0. The GIC uses this field setting to indicate that an AXI5-Stream packet is invalid.	FMU APB access error recovery on page 348
12	SM_APB_SIZE_FMU	APB size invalid. This error occurs for any FMU sparse write access, that is, the pstrb signal is not set to `0b1111`. The GIC ignores any sparse write accesses to the FMU registers.	FMU APB access error recovery on page 348
13	SM_BUSY_FMU	APB access discarded due to FMU busy error. This error occurs when FMU_STATUS.BUSY==1 and software accesses an FMU register that requires the FMU to set BUSY to 1.	FMU APB access error recovery on page 348
14	SM_BRIDGEGIC_FMU	A consistency error on the GIC-side of the GIC-FMU bridge. Asynchronous REQ/ACK error, so PROTID must be higher.	FMU reset on page 348
255	-	Software can use this setting to: • Enable or disable the error output signals on the FMU. See 6.1.6.4 Enabling or disabling both error signals on a block on page 351. • Resend any outstanding errors on the FMU. See 6.1.6.5 Discovering the active errors on a block on page 352.	-

6.1.6.1 PROTIDs reported for faults detected in LPD protection

The faults that an LPD detects are reported over a REQ/ACK interface to a GIC block in the same domain.

This GIC block determines the reported BLKTYPE (FMU error record), BLKID, and PROTID (IERR) fields in FMU_ERR<n>STATUS.

In a domain that contains multiple GIC blocks, there is a choice of where to connect LPD errors. The rendering process chooses the BLKTYPE using the following priority order: GICD, ITS, GCI, SPI Collator, Wake Request, and CPUIF. For example, if the domain includes the GICD, then the LPD errors connect to the GICD.

The reported BLKID is the BLKID of the block to which the LPD errors connect, with the exception of LPD errors that connect to a CPUIF. For LPD errors that connect to a CPUIF, the reported BLKID is the BLKID of the GCI to which the CPUIF connects.

PROTIDs for LPD errors that connect to a GICD, ITS, GCI, SPI Collator, or Wake Request

An LPD requires two PROTIDs to report errors and these depend on the chosen BLKTYPE.

The following table shows the PROTIDs that the blocks use, with the value depending on the `structure` configuration parameter.

Block type	PROTID when `structure == full` or `domain`	PROTID when `structure == wrap`
GICD	75, 76	73, 74
ITS	23, 24	21, 22
GCI	20, 21	18, 19
SPI Collator	11, 12	9, 10
Wake Request	9, 10	7, 8

See 3.8 Hierarchy on page 56 for more information about the `structure` configuration parameter.

PROTIDs for LPD errors that connect to a CPUIF block

The PROTID reported is the GCI PROTID that is allocated to that CPUIF. The PROTIDs span from 22 to 85.

On receiving a CPUIF PROTID, software can get more information by reading page-3 and page-4 of the reported PROTID. This page gives details of the reported error, including whether the error was an external error source.

6.1.6.2 Error recovery procedures

When a protection mechanism is triggered then it might be necessary for software to perform a recovery procedure, so that the GIC or system can continue functioning.

The following sections provide guidance about the recovery process that we recommend for each of the protection mechanism IDs that 6.1.6 Protection mechanism IDs on page 336 lists.

Asynchronous input error recovery

The GIC does not need to be reset after an asynchronous input error because corrupt input transitions are contained by the input protection logic. The incoming event might not have been sampled, but architectural GIC operation is unaffected.

Block reset

This reset applies to all blocks except the GICD.

The block that reported the error must be reset and also the GICD. Follow the steps in the 4.15 Power management on page 91 section to get all the blocks into the defined state before resetting them.

It is possible that the system is in a state where it is unable to complete the full powerdown sequence and a complete system reset is required.

CPUIF error recovery

Firstly, determine if the CPU interface protection error is an external error, by attempting to read the error details from page 3 and page 4 of the CPU interface protection PROTID for the connected *GIC Cluster Interface* (GCI). Page 4 provides information about deadlock protection and the error is reported when deadlock_error_seen (page 4, bit[31]) is set. This protects against faults from the CPU interface for the mixed criticality use cases.

If the error is an external error, then follow the External error recovery on page 348 procedure. Otherwise, if the error is internal to the CPUIF protection block, then reset the CPUIF, GCI, and GICD.

CRC error recovery

Read the error type from the AXI5-Stream protection block that is reporting the error. If the error type is a:

CRC timeout error

Increase the CRC timeout and continue. If the timeout is repeatable, then it might require a block reset.

CRC error

Perform a block reset, as Block reset on page 347 describes.

External error recovery

An external error does not indicate an issue within the GIC. The system integrator is responsible for ensuring that recovery is sufficient for the external error source, which might include resetting the GIC or the entire system.

FMU APB access error recovery

If the FMU has detected an incorrect access type, then there is no specific recovery procedure needed. The error has been contained by the FMU. If software was attempting an update when this error occurred, then software should repeat the update.

FMU APB recovery

If an APB parity error occurred on the register write, then perform the write again. Data in the FMU might have been corrupted.

FMU reset

Reset the FMU. Ideally the FMU Q-Channel must be quiesced before reset is applied. If resetting while not in the Q_STOPPED state, then it violates the Q-Channel protocol, so an unexpected response might be logged in the FMU when exiting reset.

Full reset

Software initiates the quiesce procedure and then resets the GIC. If the GIC fails to respond to the powerdown sequence, then a full system reset is required.

GSPV error recovery

The *GIC Stream Protocol Validator* (GSPV) protects the GCI from receiving a non-compliant protocol from a processor. The GSPV is particularly useful for mixed-criticality systems, when a GCI connects to a processor with a safety level that is lower than ASIL-D.

When GSPV reports an error, to recover the affected PE, software must reset the entire system.

Interrupt error recovery

Interrupt wire corruption can only result in either spurious or missing interrupts, and there is no need to reset the block, unless an attempt to recover the interrupt protection is required.

Interrupt errors can be due to:

- INTID-specific errors.
- Internal errors in the interrupt protection that are not associated with any specific INTID.

When recovering from an interrupt error, software must read FMU_SMRDATA page 2 for interrupt protection. If the error_INTID_valid bit returns:

0 Software can attempt a block reset for the block that contains this interrupt protection. After a block reset, if the error persists, then software can not rely on this interrupt protection.

1 Software can clear the error in page 2. If the INTID-specific error persists, software can then disable the INTID.
After disabling the INTID, if the error still persists, software can attempt a block reset for the block that contains this interrupt protection. After a block reset, if the error persists, then software can not rely on this interrupt protection.

LPD error recovery

The LPD recovery sequence depends on whether the error type as follows:

LPD error PROTID
This error requires a full reset.

LPD GIC interconnect error PROTID
This error requires a GIC reset.

Unprotected devices are AXI5-Stream interconnect components within the GIC top-level and a fault here is contained within the GIC.

P-Channel recovery

A P-Channel error can occur due to either a preq signal parity error or a pstate signal parity error. The agent driving the P-Channel should be able to move to the state it intended to, unless the preq signal has a permanent fault. If the preq signal has a permanent fault, the GIC never changes state.

Q-Channel error recovery

The GIC does not need to be reset after a Q-Channel error because corrupt transitions on the qreqn signal are contained by the Q-Channel protection logic. In the event of a qreqn signal fault, the GIC does not change its state. If in Q_STOPPED, the GIC remains there and a full reset is necessary.

If the Q-Channel error is persistent, steps should be taken to ensure that the clock that the Q-Channel controls continues to run, if necessary by keeping it running at all times.

RAM error recovery

For SECD RAM errors, no recovery procedure is required. For SECA and DED RAM errors, a recovery procedure is required. The recovery procedure is different for each type of RAM, see the following procedures:

- 4.9.4 SGI error recovery procedure on page 72
- 4.10.5 PPI error recovery procedure on page 74
- 4.11.7 SPI error recovery procedure on page 80
- 4.13.6 LPI error recovery procedure on page 89

Some RAM errors are lossy and the system integrator must determine the system recovery behavior for each RAM type. RAM error reports are also duplicated in the GICT address space.

Reset error recovery

Reset errors are contained within reset protection, so no recovery procedure is required.

Wire-only error reported recovery

When reading FMU_ERR<n>STATUS, if V=1 and W=1 but PROTID=0 (IERR=0) is observed, software should continue to read FMU_ERR<n>STATUS until either PROTID is not 0 or a timeout is reached. The timeout should be determined by the maximum AXI5-Stream fabric packet delay. If a timeout occurs, then either there is a fault with being able to receive packets from GIC blocks or there is a fault with the error wire and so the W bit reporting.

To determine if packets can be received properly, access FMU_ERRUPDATE to cause all errors to be resent for that error record. If this fails to complete normally, then perform a GIC reset recovery. If FMU_ERRUPDATE completes normally but PROTID is still 0, then there could be a fault on the error wires which set the W bit. The error wires can be disabled by clearing FMU_ERR<n>CTLR.W_EN. Operation can continue but without the error reporting redundancy provided by the error wires and the FMU_ERR<n>STATUS.W bit.

6.1.6.3 Enabling or disabling a protection mechanism

All protection mechanisms are enabled on reset.

To enable or disable a protection mechanism, write to the following fields in the FMU_SMEN register:

- BLKTYPE, selects the GIC block type
- BLKID, selects the GIC block
- SMID, selects the specific protection mechanism in the GIC block to be enabled or disabled.

We recommend that software does not disable protection mechanisms.

6.1.6.4 Enabling or disabling both error signals on a block

Each block has a critical error signal output and a non-critical error signal output. Software can enable or disable both output signals on a block.

At reset, the GIC:

- Enables the error wire outputs for the GICD and the FMU blocks.
- Disables the error wire outputs for the Wake Request and each *GIC Cluster Interface* (GCI), ITS, and SPI Collator block.

To enable or disable the block error signals, write to the following fields in the FMU_SMEN register:

- BLKTYPE, selects the type of GIC block such as GICD or ITS.
- BLKID, selects a block
- SMID, set to 255
- EN, set to:
  - 0, to disable both error signal outputs on the block.
  - 1, to enable both error signal outputs on the block.

After the GIC exits reset, we strongly recommend that the block error signals are enabled in all blocks. If block error signals are not enabled, then in the event of an error in that block, the FMU_ERR<n>STATUS.W bit is not set in the associated FMU error record. Also, a clear of that error record by writing to FMU_ERR<n>STATUS.V fails, unless FMU_ERR<n>CTLR.W_EN == 0 for that record.

How can software discover if the block error signals are enabled?

To discover if the block error signals are enabled, software can write to the FMU_SMRD register with the following fields:

- BLKTYPE, selects the type of GIC block such as GICD or ITS.
- BLKID, selects a block
- SMID, set to 255
- PAGEID, set to 0

The GIC returns the enable status of these signals when software reads FMU_SMRDATA.enable:

0 Both error wire outputs for that block are not enabled.
1 Both error wire outputs for that block are enabled.

6.1.6.5 Discovering the active errors on a block

To discover if a block has some active errors, software can write to an FMU register, to request that the block resends any errors that have not been cleared.

To request that the block resends any active errors that have not been cleared, software can write SMID=255 to any of the following registers:

- FMU_SMEN
- FMU_SMERR
- FMU_SMCR
- FMU_SMWR

When the selected block receives the message, then it resends any errors that have not been cleared.

6.1.6.6 Injecting an error in a protection mechanism

To inject an error into a protection mechanism, write to the FMU_SMERR register.

The FMU_SMERR.BLKTYPE field specifies the GIC block type, FMU_SMERR.BLKID field specifies the block ID, and FMU_SMERR.SMID field specifies the protection mechanism into which to inject the error.

When a write to FMU_SMERR completes, FMU_STATUS.BUSY remains set to 1 until any resulting updates to FMU_ERR<n>STATUS are complete. See FMU busy on page 355.

This method injects only one error. The injected errors clear when the error clears in FMU_ERR<n>STATUS.

Software can use the error injection feature to test the software error recovery handler.

6.1.7 Ping mechanisms

The GIC uses a CRC ping mechanism to protect internal AXI5-Stream connections.

The AXI5-Stream protection supports a CRC and ping mechanism that software can enable independently.

The value of the `*_stream_protection_type` configuration parameter controls the protection type for that interface as follows:

0	No protection
1	Parity protection. Suitable only for point-to-point connections.
2	CRC protection
3	CRC and parity protection

CRC protection is available only for internal AXI5-Stream interfaces, because it requires CRC hardware at both ends of the connection. If CRC protection is enabled, then a ping mechanism also runs in the background with minimal interference to "mission" traffic.

The ping mechanism is a regular check that the connection is working. Every mission packet is eventually followed by a ping packet. If a ping packet does not generate a ping acknowledge packet from the recipient, then after a timeout an error is raised. Software can program the ping timeout value in the FMU_TIMEOUT register.

An 8-bit CRC checksum is used to protect against missing or extra packets, and data corruption. The `*_sideband_crc` configuration parameter controls whether the CRC checksum is sent using either a separate sideband signal or CRC packets that the sender inserts into the stream.

CRC and ping packets try not to interfere with the mission traffic, so initially mission packets take higher priority. To prevent erroneous ping timeouts from occurring, then eventually the CRC and ping packets get a higher priority than a mission packet.

The CRC and ping mechanism adheres to the power state of the “mission” protocol, so when the bus is in low-power state, no CRC or ping packets are sent.

6.1.8 Lock and key mechanism

The FMU registers are protected against inadvertent writes by a lock and key mechanism.

The FMU registers are in a locked state after reset. If the register file is locked, then any write access to any register other than the FMU_KEY register is ignored.

The register file is unlocked when a write to FMU_KEY occurs that satisfies all of the following conditions:

- Is Secure.
- Is for 32 bits. That is, all write strobes.
- The bottom 8 bits are `0xBE`.

The register file is locked again when a write occurs that satisfies all of the following conditions:

- Is a Secure write.
- Is any width and any write strobes.
- Is to any register except for FMU_KEY.
- The write is not to the upper 32-bits of the 64-bit RAS registers, FMU_ERR<n>CTLR and FMU_ERR<n>STATUS

After a write to FMU_KEY successfully unlocks the register file, if the next access writes to FMU_KEY and:

- Does not satisfy the unlock requirements, then the register file locks.
- Satisfies the unlock requirements, then the register file remains unlocked.

If the register file is unlocked, then FMU_KEY reads as `0x00000BE`. Otherwise, FMU_KEY reads as `0x00000000`.

Non-secure accesses never succeed and never affect the locked state of the register file.

Accessing 64-bit FMU registers

Some of the FMU registers are 64-bit registers, but the APB interface width is 32 bits. When in unlocked state, the FMU allows for two consecutive writes, in any order, to update the same 64-bit register without requiring unlocking again before the second write. In this sequence, the Secure write to the upper 32 bits of the 64-bit register does not lock the FMU key, so the upper 32-bit write can either occur first or second.

For example, the following sequence is successful in updating the register contents:

1. Secure write of `0xBE` to FMU_KEY, with all write strobes asserted.
2. 32-bit Secure write to FMU_ERR0CTLR[63:32] at address offset `0x0C`, all write strobes asserted.
3. 32-bit Secure write to FMU_ERR0CTLR[31:0] at address offset `0x08`, all write strobes asserted.

This behavior is permitted to allow for the case when the APB interconnect splits a single 64-bit register access and presents it to the FMU in any order.

6.1.9 Software interaction

This section describes how software interacts with the FMU.

Initialization

The initialization routine can iterate over the FMU_ERR<n>FR registers to discover the capabilities of each error record.


Note

All protection mechanisms are enabled on reset, which might lead to errors being logged in the error records. If the system does not support or want to check a particular safety feature, then the software can disable that protection mechanism.

To disable a protection mechanism, write the corresponding block type, block ID, and protection mechanism ID to the FMU_SMEN register.

To analyze the logged errors, read the FMU_ERR<n>STATUS register.

To clear all logged errors, read the FMU_ERR<n>STATUS and write back the V, UE, and OF bits with the same value that is read.

To enable error reporting through either the ERI or CRI, write to FMU_ERR<n>CTLR.UI or FMU_ERR<n>CTLR.CI, respectively.

Interrupt handler

When an interrupt is received, the interrupt handling software identifies the error record ID by reading the FMU_ERRGSR register. The asserted bit[n] indicates that error record `n` is in error. For more information about the error, read the FMU_ERR<n>STATUS register.

FMU_ERR<n>STATUS.BLKID indicates the BLKID that reported an error, and BLKTYPE indicates the block type. FMU_ERR<n>STATUS.IERR indicates which protection mechanism reported the error.

If more than one error of the same criticality has been reported by this block type to this error record, then FMU_ERR<n>STATUS.OF is set to 1. If there is overflow, the error record retains the protection mechanism ID of the first error.

When the recovery procedure is complete, the error from this error record can be cleared by reading FMU_ERR<n>STATUS and writing back the V and OF bits with the same value that is read. The software then polls for FMU_STATUS.busy==0.

FMU busy

The APB5 port to the FMU is designed not to introduce backpressure by deasserting the pready signal. This design feature prevents software lockup and always keeps the error records accessible.

There are several operations which take multiple clock cycles to complete within the FMU. The FMU frees up the APB5 bus by asserting the pready signal to complete the APB transaction. However, it might still be processing the previous request.

When software writes to one of the following FMU registers, it must poll for FMU_STATUS.busy==0 before it issues another write to these registers:

- FMU_ERR<n>STATUS
- FMU_SMEN
- FMU_SMERR
- FMU_SMCR
- FMU_SMWR
- FMU_SMRD
- FMU_ERRUPDATE

Power management

The software can power down the Redistributor (PPI block) using the procedure that 4.15.1 Redistributor power management on page 91 describes, or the ITS could be powered down by using the GITS_CTLR register. However, the powerdown state of the PPI block and the ITS block affects certain functions of the FMU.

Writing to the following registers generates messages to the remote GIC block:

- FMU_ERR<n>STATUS
- FMU_SMEN

- FMU_SMERR
- FMU_SMCR
- FMU_SMWR
- FMU_SMRD
- FMU_ERRUPDATE

If software initiates a write to a register that targets a powered-off remote GIC block, then FMU_STATUS.BLKID_PWROFF is set to 1 when FMU_STATUS.BUSY=0.

6.2 FuSa programmer's view

The FMU contains the functional safety registers.

The GIC-700 memory map that is used to address the non-fusa legacy GIC functional logic is unchanged on GIC-720AE. See 5. Programmers model for GIC-720AE on page 132 for the functional GIC-700 memory map.

GIC-720AE uses a separate and independent memory map for the FMU programmer's view. For a description of the registers that are specific to GIC-720AE, see 5.12 FMU register summary on page 291.

6.3 FuSa I/O

The GIC-720AE has extra signals for FuSa fault detection and control.

The following table lists the protection mechanism that GIC-720AE uses for each AMBA® interface or signal type.

Table 6-9: AMBA interface FuSa ports

Interface type	Protection mechanism
APB5	AMBA parity
AXI5-Stream interfaces between internal GIC blocks	AMBA parity, or CRC, or both
Cross-chip (AXI5-Stream or ACE5-Lite)	AMBA parity, or CRC, or both
AXI5-Stream external interfaces: • CPU interface • MSI delivery interface	AMBA parity
ACE5-Lite	AMBA parity added to all external ACE5-Lite interfaces, including cross-chip when configured to use ACE5-Lite instead of AXI5-Stream.
Q-Channel	AMBA parity
P-Channel	AMBA parity

Interface type	Protection mechanism
Clock input signal	Duplicated *_chk signal
Reset input signal	Duplicated *_chk signal
Non-AMBA input signal	Odd parity *_chk signal
Non-AMBA output signal	Odd parity *_chk signal
Interrupt signal, for SPI and PPI inputs and outputs	Odd parity *_chk signals
External error interfaces	Odd parity *_chk signals

See the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual* for more information about the signals.

6.4 External error inputs

Each GIC block has generic fault inputs that allow the SoC integrator to connect and flag external faults to the FMU.

For instance, an SoC integrator might have an external safety mechanism that is physically located next to a GIC block. The SoC integrator can connect the fault signal from this external SM to the ext_err_req1 or ext_err_req0 input signals of the GIC block. If a fault occurs, the GIC flags and reports a fault in the same manner it does with internal faults.

All GIC blocks have 2 external error interfaces, but a CPUIF block can have 1-8 external error interfaces, depending on the setting of the `NUM_EXT_ERR_IF` build-time option.

A captured fault is reported in the GIC block error record with a SM_EXT<n>_<BLKTYPE> protection name such as SM_EXT0_GICD or SM_EXT1_ITS.

6.5 Resets

GIC-720AE uses active-LOW duplicated resets. The two duplicated resets must change on the same clock edge, although GIC-720AE has an allowed skew tolerance after synchronizing the two resets, before a reset error is generated.

GIC-720AE only enters reset when both duplicated resets are asserted.

GIC-720AE exits reset when either duplicated reset deasserts.

The GIC-720AE has a reset synchronizer, so that on reset the post-synchronizer reset signals assert asynchronously and deasserts synchronously.

Although the FMU resides in the GICD domain, the FMU has separate fmu_reset_n and fmu_reset_n_chk signals. Therefore, the GIC can be reset without resetting the FMU, which enables retention of the FMU error records.

6.5.1 Cold reset sequence

Follow these steps to initiate a Cold reset of the GIC.

Procedure

1. Assert either reset_n and reset_n_chk, or dbg_reset_n and dbg_reset_n_chk signals simultaneously.
   - The post-synchronizer reset signals, reset_n_pri and reset_n_sec, assert asynchronously at the same time.
2. Keep the resets asserted for at least 3 cycles more than the number of stages in the reset synchronizer.
   Step result: This reset assertion guarantees a reset flush through the non-resettable flops.
3. Release the resets.
   When either:
   - The reset_n or reset_n_chk signal deasserts, the reset_n_pri signal deasserts synchronously, followed by the reset_n_sec signal two clk cycles later. reset_n and reset_n_chk signals deassert at the same time.
   - The dbg_reset_n or dbg_reset_n_chk signal deasserts, the dbg_reset_n_pri signal deasserts synchronously, followed by the dbg_reset_n_sec signal two clk cycles later. dbg_reset_n and dbg_reset_n_chk signals deassert at the same time.

6.5.2 Warm reset sequence

Follow these steps to initiate a Warm reset of the GIC.

About this task

A Warm reset is a reset that occurs after the component has already been operating for some time. A Warm reset preserves the state of the PMU and the FMU error records, in both the functional and FuSa GIC address maps. This state preservation is accomplished by not toggling the dbg_reset_n signals. Before resetting a GIC block, the quiescing procedure must be followed for the block being reset. This procedure ensures that the reset can be performed cleanly.

Procedure

1. Assert the reset_n and reset_n_chk signals simultaneously.
   The resynchronized reset signals, reset_n_pri and reset_n_sec, assert asynchronously at the same time.
2. Keep the resets asserted for at least 3 cycles more than the number of stages in the reset synchronizer.
   Step result: This reset assertion duration guarantees a reset flush through the non-resettable flops.
3. Release the resets.
   When either the reset_n or reset_n_chk signal deasserts, the reset_n_pri signal deasserts synchronously, followed by the reset_n_sec signal two clk cycles later.

6.6 Lock-step protection

The GIC-720AE logic is protected by redundant lock-step checking.

The exceptions to this are:

- The RAMs, which are shared.
- The internal AXI5-Stream interconnect, which uses CRC for end-to-end protection.
- Real-time interrupts.

Lock-step has a temporal delay of two cycles.

The entire `noram` hierarchy is duplicated, with the comparators instanced in the block top level. The clock gate and reset synchronizers are also duplicated in the top level.

The clocking is also duplicated. To provide redundancy in the reset and clock trees, the primary and secondary logic are clocked by a separate clock and have a separate reset. In the clock tree, if a branch of the reset fails in the primary domain, then the secondary domain detects the failure. Similarly, if a branch of the reset fails in the secondary domain, then the primary domain detects the failure.

Lock-step protection is distributed into all protection mechanisms that connect to primary and secondary GIC `noram` output ports. Therefore, all outputs from the `noram` that become outputs of the GIC block, are checked using local lock-step checkers within the associated interface protection mechanism. The lock-step checking logic consists of a 2-cycle delay stage for the primary signals followed by duplicated comparators for latent fault protection. The lock-step comparison is performed per input bit in all checker instances, except for the MBIST read data checker that performs a comparison of an 8-bit CRC checksum of the incoming wide data.

6.7 RAM protection

RAM instances are not duplicated and are shared by both primary and secondary `noram` instances.

RAMs are protected with the GIC ECC scheme that distinguishes between:

- SEC errors on the data.
- SEC errors on the address.
- DED errors.
- White noise errors, that is, detection of all 0s or all 1s data.

For each RAM, the fault collator allocates a different protection mechanism ID for these RAM error types. Therefore, for a block with many RAMs such as GICD, the RAM errors are a significant proportion of the protection mechanism ID space.

RAM protection also performs lock-step checking of the RAM address, data, and control outputs for the primary and secondary logic.

6.8 MBIST data save and restore

The system integrator must ensure that the MBIST controller checks the MBIST read data integrity, if the controller uses that data to restore memory state.

This means that the integrity of the data must be guaranteed during its capture and storage, avoiding all single points of failure and written back to the GIC from independent sources that again avoid single points of failure.

It is the system integrators responsibility to decide how to achieve this data integrity. One approach could be to capture the mbistoutdata and mbistoutdata_chk signals and store them independently in two separate memories. When restoring the RAM data, the two memories would then be used to drive data onto the mbistindata and mbistindata_chk signals independently. Any inconsistencies between mbistindata and mbistindata_chk would be detected by the GIC within MBIST protection.

6.9 Interrupt protection BIST and the DetectionPaused bits

After reset, before software programs the interrupt protection page-1 control registers, we recommend that software waits for interrupt protection BIST to complete. When the BIST completes, the GIC sets the page-1 FMU_SMRDATA.BISTBusy bit to 0.

During interrupt protection BIST, the GIC temporarily disables the error detection for all INTIDs. When page-1 FMU_SMRDATA.DetectionPaused is set to 1 for an individual INTID, error detection is temporarily disabled for that INTID. The page-2 FMU_SMRDATA.DetectionPaused is the logical OR of all page-1 DetectionPaused bits, indicating that one or more INTIDs have page-1 DetectionPaused set to 1.

When BIST completes, that is FMU_SMRDATA.BISTBusy == 0, software must then read the page-2 FMU_SMRDATA.DetectionPaused bit. When page-2 FMU_SMRDATA.DetectionPaused == 0, this value indicates that all interrupt protection logic has been able to reach a state where error detection can begin. A permanent fault on an interrupt wire or its corresponding check wire, prevents the page-2 DetectionPaused bit from being set to 0.

If page-2 DetectionPaused remains set to 1, software can:

1. Read page-1 for all INTIDs, to discover which INTIDs still have FMU_SMRDATA.DetectionPaused == 1.
2. Disable the interrupt protection for that INTID, by writing 0 to the page-1 FMU_SMRDATA.Enable bit, which causes page-1 FMU_SMRDATA.DetectionPaused to become 0.
3. Repeat step 2 for any other INTIDs where page-1 FMU_SMRDATA.DetectionPaused == 1.
4. Verify that page-2 FMU_SMRDATA.DetectionPaused == 0.

Appendix A Getting started with GIC-720AE

There are some basic tasks that you must complete before you can start to use GIC-720AE.

Each Redistributor must be powered on using its GICR_PWRR register to enable the Redistributors to be accessed, see 4.15.1 Redistributor power management on page 91 for more information.

When the GIC-720AE is powered up, it must be programmed as the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4* describes.

A.1 Removing cores from a preconfigured GIC

The GIC can be configured to either enable Secure software or a tie-off signal to remove cores from a GIC configuration. This feature enables you to use a single GIC configuration in multiple products that contain a different number of cores.

The `prog_mpidr` configuration parameter controls whether software or hardware can remove cores from a GIC configuration.

Software control, when `prog_mpidr == prog`

This `prog_mpidr` setting enables Secure software to remove cores during the boot up of a system. If GICD_CTLR.DS == 1, then Non-secure software can remove cores. If GICD_CFGID.VIEW == 1, then only view 0 has access to perform the necessary changes. The software flow is:

1. Software checks if GICD_CFGID.RDC == 1. When set to 1, it confirms that software can remove cores from the configuration.
2. Software writes to GICD_RDOFFR`n` and sets a bit to 1 to remove that core from the configuration. `n` has a value of 0-7 and each value represents 64 cores. For example, to remove:
   - The 1st core, set GICD_RDOFFR0[0] to 1.
   - The 22nd core, set GICD_RDOFFR0[21] to 1.
   - The 72nd core, set GICD_RDOFFR1[7] to 1.

   When cores are removed, the affinity values of the remaining cores automatically change, so software must then program GICR_MPIDR. See Requirement to program GICR_MPIDR on page 362.
3. Software writes to each GICR_MPIDR to set the affinity values for the cores on that Redistributor. The address map for these Redistributors is now a single contiguous block of Redistributor address space.
4. Software can then start normal operation.


Note

Software must program the GICD_RDOFFRn and GICR_MPIDR registers before any other GIC registers are accessed (other than reads to GICR_TYPER and read-only ID registers) and before the GIC receives messages from any cores. Otherwise the behavior is unpredictable.

Example A-1: Requirement to program GICR_MPIDR

When software uses GICD_RDOFFRn to remove a core, the following core in the sequence then effectively inherits the affinity settings of the removed core. The following example shows the importance of the subsequent programming of the GICR_MPIDR registers.

In this example, there are 4 Redistributors with the following affinity values:

Redistributor 0 0.0.0.0, physical PE 0
Redistributor 1 0.1.0.0, physical PE 1
Redistributor 2 0.2.0.0, physical PE 2
Redistributor 3 0.2.1.0, physical PE 3

If software writes `0x2` to GICD_RDOFFR0, it removes PE 1 and its Redistributor, and the affinity values for the remaining Redistributors are:

Redistributor 0 0.0.0.0, physical PE 0
Redistributor 1 0.1.0.0, physical PE 2
Redistributor 2 0.2.0.0, physical PE 3

The original Redistributor 2 and Redistributor 3 are now in separate clusters, but previously they were in the same cluster. Therefore, to retain the intended heirarchy, software must also program the GICR_MPIDR registers.

Hardware control, when `prog_mpidr == strap`

This `prog_mpidr` setting enables hardware to remove cores as the GIC exits reset. With this option, the software is unaware that the GIC is supporting fewer cores than the configuration allows.

This option provides the following extra tie-off signals:

gicd_pe_off[max_pe_on_chip – 1:0]
Set a bit to 1, to remove the corresponding core. The behavior is unpredictable when all bits are set to 1.

affinity0[(max_pe_on_chip × max_affinity_width0) – 1:0]
Sets the affinity 0 value for each core.

affinity1[(max_pe_on_chip × max_affinity_width1) – 1:0]
Sets the affinity 1 value for each core.

affinity2[(max_pe_on_chip × max_affinity_width2) – 1:0]
Sets the affinity 2 value for each core.

affinity3[(max_pe_on_chip × max_affinity_width3) – 1:0]

Sets the affinity 3 value for each core.

Note

These tie-off signals must be set before the GIC is taken out of reset and must remain stable, otherwise the behavior is unpredictable. If the width of the signal is zero, then it is not present on the GIC instance.

The bit order in these tie-off signals is the order that the Redistributor pages appear in the default GIC address map, as defined by the order of GCI blocks and buses within them. These values are set by the `ppi_ref` and `bus` parameters in the configuration file, that is, there is a fixed relationship between the tie-off signal and a physical processor.

Example A-2: Example of removing cores from a 4-core configuration

This 4-core example has affinity 0, 1, and 2 with a width of 2 bits:

Core 0 MPIDR 0.0.0.0
Core 1 MPIDR 0.0.0.1
Core 2 MPIDR 0.0.1.0
Core 3 MPIDR 0.0.1.1

The following table shows the tie-off signal values when core 1 is removed and also when core 0 and core 2 are removed.

Signal	No cores removed	Core 1 removed	Core 0 and 2 removed Core 1 in each cluster moved to 0
gicd_pe_off	0b0000	0b0010	0b0101
affinity0	0b01_00_01_00	0b01_00_xx_00	0b00_xx_00_xx
affinity1	0b01_01_00_00	0b01_01_xx_00	0b01_xx_00_xx
affinity2	0b00_00_00_00	0b00_00_xx_00	0b00_xx_00_xx

When cores are removed by setting bits of the gicd_pe_off signal, the GICD updates other software-visible features so that software cannot detect the reduced core count. These updates include:

- Moving GICR_TYPER.Last to the last Redistributor.
- Moving the GICDA register page to the page above the last Redistributor.

Limitations

The removal of cores from a configuration, by software or hardware, has the following limitations:

GICR_CFGID0.PPI_number

This field reflects a tie-off on the *GIC Cluster Interface* (GCI). The system integrator must change the tie-off as required. The tie-off has no function other than implementation-defined discovery, so the tie-offs could all be tied to the same value.

FMU

The removal of cores does not change the protection mechanism mappings in the FMU. Therefore, the firmware that accesses the APB interface must know the full structure of the GIC configuration, especially if all cores on a particular GCI or CPUIF protection block are removed.

MBIST

The GIC does not alter the MBIST interface, so the system integrator must add any protection that is required.

Removed cores

If cores are removed, then the behavior is unpredictable if the GIC receives GIC Stream messages from a removed core.

GICR<n>_ERRINSR

These registers are used for inserting errors, so that software can check the ECC operation on the RAMs in the GCI block.
However, if cores are removed then these registers are not updated. Therefore, when some, but not all, cores are removed from a cluster interface, the GIC reports errors only in the RAS records of the available cores. This behavior provides a mechanism for software to determine which cores are removed.

A.2 Other power management

The GIC-720AE can be powered up and powered down using non-architectural protocols.

When powering up GIC-720AE, then software must program registers in the following sequence:

1. If using programmable core removal, program GICD_RDOFFRn and then GICR_MPIDR.
2. If using multi view, program GICR_VIEWR.
3. Any other registers.

When powering down GIC-720AE, software must preserve the state of the GIC-720AE, except for any LPI pending interrupts that are preserved in pending tables, as defined in the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

You can preserve the LPI pending bits by using an implementation-defined powerdown sequence, which ensures that the memory pointed to by each GICR_PENDBASER contains the updated pending information for the LPIs. The implementation-defined powerdown sequence must:

1. Complete the powerdown sequence for all cores.
2. Set GICR_WAKER.Sleep to 1.
3. If GICD_TYPER.LPIS==1, poll GICR_WAKER until GICR_WAKER.Quiescent is set.


Note

- GICR_WAKER.Sleep can only be set to 1 when:
  - All Redistributors have GICR_WAKER.ProcessorSleep == 1.
  - All Redistributors have GICR_WAKER.ChildrenAsleep == 1.
- GICR_WAKER.ProcessorSleep can only be set to 0 when:
  - GICR_WAKER.Sleep == 0.
  - GICR_WAKER.Quiescent == 0.
- If software decides to abort a sleep request due to an external wake request, it can do so by clearing GICR_WAKER.Sleep at any time. Software does not have to wait for GICR_WAKER.Quiescent to be set.
- There is only one GICR_WAKER.Sleep and one GICR_WAKER.Quiescent bit that can be read and written through the GICR_WAKER register of any Redistributor.

The powerdown described sequence ensures that all LPIs that are acknowledged by a write response to the write GITS_TRANSLATER are saved to the Pending tables. Any interrupt that arrives when the Sleep bit is set to 1 is ignored, and the ACE5-Lite transaction completes in accordance with the ACE protocol.

We recommend that you disable any interrupt sources before setting GICR_WAKER.Sleep. However, if you require wake-on-interrupt behavior, the write to GITS_TRANSLATER must be gated upstream at a location that enables software to reprogram and enable the GIC-720AE without deadlock.

When the GICR_WAKER.Quiescent bit is set, it is safe to power down the GIC-720AE without losing LPI pending bits. Software must still perform other steps such as the save and restore of SPI state. However, you must provide custom mechanisms to wake the GIC-720AE if any interrupts arrive that must not be ignored.

When the GIC-720AE next powers up, you can program the GICR_PENDBASER registers to point to the same memory to reload the LPI pending status. If there is no requirement to reload the pending LPIs, we recommend that you speed up the initialization of the GIC-720AE as follows:

1. Zero the Pending table.
2. Set GICR_PENDBASER.PTZ to 1.


Note

GICR_PENDBASER registers can only be modified before the GICR_CTLR.Enable_LPIs bit is set, or when the GICR_WAKER.Sleep and GICR_WAKER.Quiescent bits are both set.

For more information, see the *Learn the architecture - Arm® Generic Interrupt Controller v3 and v4*.

Related information

GICR_WAKER, Power Management Control Register on page 195

A.3 Setting error recovery and fault handling options

Use the following procedures to set the error recovery and fault handling option.

Procedure

1. Write to GICT_ERR<c>MISC0.Count to preset the counter to any value.
   For example, to fire an interrupt on any correctable error, write `0xFF`, or to fire an interrupt on every second correctable error, write `0xFE`.
2. Assign a recorded uncorrectable ECC error to one of these options:
   - The fault-handling interrupt, fault_int signal, by setting GICT_ERR<n>CTLR.FI.
   - The error recovery interrupt, err_int signal, by setting GICT_ERR<n>CTLR.UI. The interrupt fires on every uncorrectable interrupt occurrence irrespective of the counter value.

   We recommend that if the err_int and fault_int signals are internally routed, the target interrupts must not have SPI Collator wires, or if they are present, are tied off. This prevents software checking for the same ID at multiple destinations. The err_int and fault_int signals do not have direct test enable registers. You can test connectivity using error record 0 and triggering an error, such as an illegal AXI access to a nonexistent register.
3. Route the fault_int and err_int output signals as either:
   - Interrupt wires for situations where error recovery is handled by a core that does not receive interrupts directly from the GIC, such as a central system control processor.
   - Drive each interrupt internally by programming the associated GICT_ERRIRQCR<n> register. Each GICT_ERRIRQCR<n> register contains an ID field that must be programmed to 0 if internal routing is not required, or if internal routing is required, to a legally supported SPI ID.


Note

If the programmed ID value is less than 32, out of range, or not owned on chip for multichip configurations, the register updates to 0 and no internal delivery occurs.

A.4 Setting a PMU counter

Use the following procedure to configure a counter.

About this task


Note

PMU registers, other than enables, do not have defined reset values and must be programmed before use.

Procedure

1. Program the counter GICP_EVCNTRn to a known value. This value could be 0 to count events, or a higher number to trigger an overflow after a known number of events.
2. Program the associated GICP_EVTYPERn to count the required event.
3. Program the required filter type for the event by programming GICP_FRn.
4. Enable the counter by programming the corresponding bit in GICP_CNTENSET0.
5. Repeat the previous steps for all counters that are required.
6. Enable the global count enable in GICP_CR.E.

A.5 Connecting the chips

Use the following procedure to connect the chips in a multichip configuration.

Before you begin

The following restrictions apply when connecting or removing chips:

- You must consider that data that is read from GICD_CHIPRn is valid only when GICD_DCHIPR.PUP == 0, otherwise the data might be updating.
- If you are connecting a new chip, the accesses must be done through a chip that is in the Consistent state and not by writing to the new chip directly.
- If you access GICD_CHIPSR while a chip is being connected, it shows RTS == Updating. Also, the GICD_DCHIPR.PUP bit is set, indicating that the Routing table is updating, so the values cannot be trusted.
- Adding or removing a chip when GICD_CTLR group enables are set is unpredictable. To check that group enables are off, software must poll GICD_CTLR.RWP.
- If you are connecting together multiple different instances of the GIC-720AE, the settings of the gicd_ctlr_ds signal must match in all chips.
- If you are connecting together multiple different instances of the GIC-720AE, the settings for the following parameters must match in all chips:
  - All affinity widths (`max_affinity_width`*)
  - Number of SPI blocks supported (`spi_blocks`)
  - LPI support type (`lpi_support`)

- Total number of chips supported (`chip_count`)
- Chip address width (`chip_addr_width`)
- Chip affinity select level (`chip_affinity_select_level`)
- Maximum number of cores on any single chip (`max_pe_on_chip`)
- The number of vPEs (`vpe_width`)
- GICv4.1 architecture support (`gicv41_support`)
- 1 of N support (`spi_1ofn_support`)
- Cross-chip interface protocol (`ace_cc`)
- Cross-chip addressing mode (`local_chip_addr`)
- Non-maskable interrupt (NMI) support (`nmi_support`)
- Multi view support (`multi_view_support`)

See the *Arm® CoreLink™ GIC-720AE Generic Interrupt Controller Configuration and Integration Manual* for information about configuration parameters and their options.

About this task

The procedure for connecting the chips in a multichip configuration is as follows:

Procedure

1. Ensure that the values of the chip_id tie-off input signals to all chips are correct.
2. Ensure that all Group enables in the GICD_CTLR register are disabled and GICD_CTLR.RWP == 0.
3. Designate a chip, chip `x`, to own the Routing table.
   You can designate a different chip later, if necessary.
4. Before software brings a chip online by writing to the RT owner, it must program all local GICD_CHIPRn.ADDR fields. The procedure depends on whether local chip addressing is enabled.
   When GICD_CFGID.LCA == 0:

   a) Software programs all the GICD_CHIPRn.ADDR fields from a single chip, ideally the RT owner, by writing to a single Distributor instance.
      When the chip comes online, it broadcasts the address values to the other chips.

   When GICD_CFGID.LCA == 1:

   a) Software programs all the GICD_CHIPRn.ADDR fields, for each chip separately.
      For example, each chip writes all the required GICD_CHIPRn.ADDR values to its own Distributor. Software must ensure that all remote chip addresses are unique from any given chip.

      When the chip comes online, the address values are not broadcast to the other chips.
5. In a single register write, program GICD_CHIPRx with:
   a) GICD_CHIPRx.ADDR so that each chip can forward messages to chip `x`.

Depending on how cross-chip messages are routed, this value can be the chip_id signal value or a more complex identifier. For an:

- AXI5-Stream cross-chip interface, this value is sent on the icdrtdest signal.
- ACE5-Lite cross-chip interface, this value is sent on the awaddr[AXIM_ADDR_WIDTH −1:16] signal.

b) GICD_CHIPRx.SPI_BLOCK_MIN and GICD_CHIPRx.SPI_BLOCKS to appropriate values for the SPIs that chip `x` owns.
Step example: If the range of interrupt ids for chip `x` is ID96-ID159:

- Set SPI_BLOCK_MIN = (96 – 32) / 32 = 2
- Set SPI_BLOCKS = (159 – 96 + 1) / 32 = 2

c) GICD_CHIPRx.SocketState = 1

6. To check that the writes are successful, read GICD_CHIPRx.
The writes might fail due to security settings, an overlapping or nonexistent SPI, or if another update is still in progress. If the accesses fail, then GICD_CHIPRx.SocketState == 0, indicating that the chip is offline.
7. To check that the actions of this sequence have executed correctly, read the following register fields and ensure that their values are as follows:

   a. GICD_CHIPSR.RTS == 2 (Consistent)

   b. GICD_DCHIPR.rt_owner == chip `x`

   c. GICD_DCHIPR.PUP == 0

   Step result: Chip `x` is now in the Consistent state and ready to accept connections to other chips in the system configuration.

To connect more chips:

8. Set the relevant address and SPI ownership information of the next chip you want to connect to, chip `y`, by writing to GICD_CHIPRy.
You can do this step through any chip that is already connected, or more efficiently by writing to the chip that owns the Routing table, chip_id signal value == rt_owner.
9. Poll GICD_DCHIPR until bit PUP == 0, indicating that the connection is complete.
10. To check whether the write to GICD_CHIPRy is accepted, read GICD_CHIPRy.
For each chip connection, repeat steps 8 on page 369 through 10 on page 369.

A.6 Changing the Routing table owner

In a multichip system, you can change the chip that owns the Routing table at any time. However, the Routing table owner must be the last chip to be powered down.

About this task

The following procedure describes how to change the owner of the Routing table:

Procedure

1. Write to GICD_DCHIPR.rt_owner with a value that selects the appropriate chip to be the Routing table owner.
   The chip_id signal sets the identification value of a chip.
2. Poll for GICD_DCHIPR.PUP == 0.

A.7 Isolating a chip from the system

In a multichip system, you can isolate a chip from the system.

About this task

To isolate a chip from the system, use the following procedure:

Procedure

1. Ensure that all cores on the chip are asleep by setting GICR_WAKER.ProcessorSleep.
2. Ensure all ITS blocks on the chip are disabled and the buses are quiesced by using the qreqn_its<n> Q-Channel interfaces.
   Before isolating the chip, the ITSs must be powered off because the Routing table is invalid when the GIC P-Channel is in the OFF state.
3. Ensure that LPIs from other chips are not routed to this chip.
4. Attempt to enter the CONFIG state (pstate signal = `0x9`).
   If the GIC is idle and all credits are returned, it accepts the request to go into CONFIG state, otherwise it denies the request and remains in RUN state.

   
   Note

   All SPIs must return to their own chip before a request is accepted. This means that SPIs that are enabled and pending, but targeting a core on a remote chip where the relevant CPU group is disabled, prevent transition into the CONFIG state.

   When in the CONFIG state, any cross-chip messages that change the internal state are held in the cross-chip interface, and all messages assert the pactive signal. If the pactive signal asserts while attempting to enter a lower power state, you must return to RUN (pstate signal == `0x0`).
5. When in CONFIG state, any required state can be saved.
   Writing to GICD_CHIPRn or GICD_DCHIPR for any purpose other than to restore saved values after a hardware reset is unpredictable.
6. If using GICv4.1, then software must write and poll the GICR_VINVCHIPR register on at least one PE from all the other chips. This check ensures that no stale cached vPE routing information exists that would unnecessarily wake the chip that is being powered down.
7. Power down the Redistributors using the GICR_PWRR registers.
8. Optional: Flush the LPI cache using GICR_WAKER.Sleep.
   We recommend that if wake-on-interrupt is required, LPIs from other chips do not target this chip while the chip is being powered down (step 3 on page 370). Also, LPIs from other chips must be routed back while the chip is in the OFF state.

If LPIs arrive after sleep is set in the CONFIG state, then the LPIs are dropped.

9. Attempt to enter the OFF state.
   If the pactive signal is HIGH, return to the CONFIG state.
10. Use the Q-Channel to put the GIC into a safe mode to reset.
    If the SPI Collator is in a different domain to the Distributor and only one of the domains is being reset, then the power Q-Channel must have also accepted before the reset can occur. This might require masking interrupts outside of the GIC to ensure that all interrupt lines have reached their idle state.

    Power up is the reverse of the powerdown sequence. However, you must ensure that the Routing table is restored before other registers, else the behavior is unpredictable. Restoring values to the Routing table that are not exactly the same as those values read out before a reset, can cause unpredictable behavior.


Note

Accesses to GICD_CTLR continue to be broadcast to the isolated chip, which requests wakeup.

Related information

Power control and P-Channel on page 93

Appendix B Implementation-defined features of GIC-720AE

The GIC-720AE implements features that are defined in the GICv4.1 architecture. Many of these features also have options in the GICv4.1 architecture, which determine behavior that is specific to the GIC-720AE. These features and options are configurable at build time.

The following table summarizes the implementation-defined features of the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* that GIC-720AE uses. The table also gives references to sections within this manual that provide information about implementation-defined behavior that is specific to the GIC-720AE.

Table B-1: Declared implementation-defined features

GICv4.1 architecture feature	Architectural specification reference		Description
	Chapter	Section	
1 of N model	Introduction	Models for handling interrupts	See 4.11.3 SPI routing and 1 of N selection on page 77
Direct LPI support	GIC partitioning	The GIC logical components	Direct LPI support, that is, using the GICR_SETLPIR and GICR_CLRLPIR registers, is not supported.
ITS to Redistributor communications	Locality-specific peripheral interrupts and the ITS	LPIs	This communication occurs over a fully credited AXI5-Stream.
INTIDs	Distribution and routing of interrupts	INTIDs	16-bit width when supporting LPIs, otherwise the width is set to support the number of SPIs and SGIs.
All error cases	-	Pseudocode throughout the document	All errors are reported through error records, see 4.17 Reliability, Accessibility, and Serviceability on page 97.
Message-based SPIs	Physical interrupt handling and prioritization	Shared peripheral interrupts	Pending bits for level sensitive SPIs that are set by writes to GICD_SETSPI_* or GICM_SETSPI_* are not affected by writes to GICD_ICPENDRn. Writes to GICD_CLRSPI_* or GICM_CLRSPI_* have no effect on pending bits set by GICD_ISPENDRn.
Interrupt grouping	Physical interrupt handling and prioritization	Interrupt grouping	All implemented SPIs, SGIs, and PPIs have programmable groups.
Interrupt enables	Physical interrupt handling and prioritization	Enabling individual interrupts	All SGIs have a programmable enable.
Interrupt prioritization	Physical interrupt handling and prioritization	Interaction of group and individual interrupt enables	Interrupts that are disabled through the GICC_CTLR register or the ICC_CTLR_* registers are not considered in the selection of the highest pending interrupt and do not block fully enabled interrupts of a lower priority.

GICv4.1 architecture feature	Architectural specification reference		Description
	Chapter	Section	
		Interrupt prioritization	GIC-720AE supports 32 priority levels, 16 for LPIs that are always Non-secure.
Effects of disabling interrupts	Physical interrupt handling and prioritization	Effect of disabling interrupts	Interrupts are set pending irrespective of the GICD_CTLR.EnableGrp* settings.
Changing priority	Physical interrupt handling and prioritization	Interrupt prioritization. Changing the priority of enabled PPIs, SGIs, and SPIs.	Reprogramming an IPRIORITYRn register does not change the priority of an active interrupt but causes a pending and not active interrupt to be recalled from the CPU interface so that the new priority value can be applied.
LPI caching	Locality-specific peripheral interrupts and the ITS	LPIs	See 4.13.4 LPI caching on page 88 and 4.12 ITS on page 82.
LPI Configuration tables	Locality-specific peripheral interrupts and the ITS	LPI Configuration tables	The GIC-720AE has one GICR_PROPBASER register for all cores on a chip and therefore points to a single table. Each chip in a multichip configuration can point to a copy of the table in local memory. See GICR_TYPER.CommonLPIAff for more information. When interrupts are sent between chips, they keep the properties associated with them until the next invalidate. All property fetches are always from the offset specified in the GICR_PROPBASER register of the issuing chip.
LPI Pending tables	Locality-specific peripheral interrupts and the ITS	LPI Pending tables	See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*

Appendix C Revisions

This appendix describes the technical changes between released issues of this document.

Table C-1: Issue 0000-01

Change	Location
First release	-

Table C-2: Differences between issue 0000-01 and issue 0001-02

Change	Location
Corrected gicd_ctrl_ds to gicd_ctlr_ds. Corrected gicd_ctrl_ds_chk to gicd_ctlr_ds_chk.	4.3 Interrupt groups and security on page 59
Added the Revision value for r0p1.	• 5.2.3 GICD_IIDR, Distributor Implementer Identification Register on page 145 • 5.3.2 GICM_IIDR, Message-based Distributor Implementer Identification Register on page 187 • 5.4.2 GICR_IIDR, Redistributor Implementation Identification Register on page 192 • 5.7.1 GITS_IIDR, ITS Implementer Identification Register on page 230 • 5.10.8 GICT_IIDR, Trace Implementer Identification Register on page 265 • 5.11.14 GICP_IIDR, PMU Implementer Identification Register on page 288
Added the Version value for r0p1.	5.5.10 GICR_CFGID1, Configuration ID1 Register on page 217
For CPU interface protection, added information about instability in the iritdest_*_crc signal.	5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
Corrected gicd_ctrl_ds to gicd_ctlr_ds.	A.5 Connecting the chips on page 367

Table C-3: Differences between issue 0001-02 and issue 0100-03

Change	Location
Added multi view feature and its associated registers.	• 4.5 Multi view on page 63 • 5.2.29 GICD_ICVERRRn, Interrupt Clear View Error Registers on page 177 • 5.2.30 GICD_ICVERRRnE, Interrupt Clear View Error Registers Extended on page 178 • 5.2.31 GICD_IVIEWRn, Interrupt View Registers on page 179 • 5.2.32 GICD_IVIEWRnE, Interrupt View Registers Extended on page 180 • 5.4.10 GICR_VIEWR, View Register on page 202

Change	Location
Added the Variant value for r1p0.	• 5.2.3 GICD_IIDR, Distributor Implementer Identification Register on page 145 • 5.3.2 GICM_IIDR, Message-based Distributor Implementer Identification Register on page 187 • 5.4.2 GICR_IIDR, Redistributor Implementation Identification Register on page 192 • 5.7.1 GITS_IIDR, ITS Implementer Identification Register on page 230 • 5.10.8 GICT_IIDR, Trace Implementer Identification Register on page 265 • 5.11.14 GICP_IIDR, PMU Implementer Identification Register on page 288
Added the Version value for r1p0.	5.5.10 GICR_CFGID1, Configuration ID1 Register on page 217
Added a register for GSPV.	5.4.11 GICR_FLUSHR, Flush Register on page 203
Corrected the `number_ll_int_credit` parameter name.	5.7.11 GITS_CFGID, Configuration ID Register on page 246
Add information about page accesses to PROTIDs that are not present in the configuration.	• 5.12.10 FMU_SMWDATA, Safety Mechanism Write Data register on page 309 • 5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
For CPU interface protection, an instability in the iritdest_*_crc signal sets the crc_chksum_err bit.	5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
Corrected the number of CPUIF protection blocks from 64 to 16.	6.1.6 Protection mechanism IDs on page 336
Corrected the description of the Direct LPI support feature.	B. Implementation-defined features of GIC-720AE on page 372

Table C-4: Differences between issue 0100-03 and issue 0001-04

Change	Location
For the SMID field, added information about setting EN and SMID=255.	5.12.6 FMU_SMEN, Safety Mechanism Enable register on page 303
Corrected the bit names and description for the AXI5-Stream cross-chip protection.	• 5.12.10 FMU_SMWDATA, Safety Mechanism Write Data register on page 309 • 5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
Added information about page 1 timeouts for ACE5-Lite cross-chip protection.	• 5.12.10 FMU_SMWDATA, Safety Mechanism Write Data register on page 309 • 5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
Added a restriction for the SMID field when BLKID_ERR == 1 or BLKID_PWROFF == 1.	5.12.13 FMU_STATUS, FMU Status Register on page 321

Table C-5: Differences between issue 0001-04 and issue 0200-05

Change	Location
Added support for real-time interrupts.	4.11 SPIs on page 75

Change	Location
Updated the description about reassigning SPIs with GICD_IVIEWRn.	4.5 Multi view on page 63
Added information about the excessive assertion of the pmu_int signal.	Overflow interrupt on page 96
Changed GICT_ERR1MISC0 to GICT_ERR<n>MISC0.	4.17.4.2 SPI RAM error records 1-2 on page 105
Updated the SPF bit description.	5.2.6 GICD_SAC, Secure Access Control register on page 149
Updated the CC_SHARED bit description.	5.2.12 GICD_CCCTLR, Cross-Chip Control Register on page 157
Updated the SPI_BLOCK_MIN and SPI_BLOCKS descriptions.	5.2.15 GICD_CHIPR<n>, Chip Registers on page 160
Added the GSPV_CGO bit.	5.4.7 GICR_FCTLR, Function Control Register on page 198
Updated the AccessType bit description.	5.5.1 GICR_MISCSTATUSR, Miscellaneous Status Register on page 209
Added the Variant value for r2p0.	• 5.2.3 GICD_IIDR, Distributor Implementer Identification Register on page 145 • 5.3.2 GICM_IIDR, Message-based Distributor Implementer Identification Register on page 187 • 5.4.2 GICR_IIDR, Redistributor Implementation Identification Register on page 192 • 5.7.1 GITS_IIDR, ITS Implementer Identification Register on page 230 • 5.10.8 GICT_IIDR, Trace Implementer Identification Register on page 265 • 5.11.14 GICP_IIDR, PMU Implementer Identification Register on page 288
Added the Version value for r2p0.	5.5.10 GICR_CFGID1, Configuration ID1 Register on page 217
Updated the OF bit description.	5.10.3 GICT_ERR<n>STATUS, Error Record Primary Status Register on page 253
Clarified the behavior of successive writes to the FMU_KEY register.	6.1.8 Lock and key mechanism on page 353
Added a limitation for the FMU.	A.1 Removing cores from a preconfigured GIC on page 361

Table C-6: Differences between issue 0001-04 and issue 0001-06

Change	Location
Added information about the excessive assertion of the pmu_int signal.	Overflow interrupt on page 96
Changed GICT_ERR1MISC0 to GICT_ERR<n>MISC0.	4.17.4.2 SPI RAM error records 1-2 on page 105
Updated the SPI_BLOCK_MIN and SPI_BLOCKS descriptions.	5.2.15 GICD_CHIPR<n>, Chip Registers on page 160
Updated the OF bit description.	5.10.3 GICT_ERR<n>STATUS, Error Record Primary Status Register on page 253
Clarified the behavior of successive writes to the FMU_KEY register.	6.1.8 Lock and key mechanism on page 353
Added a limitation for the FMU.	A.1 Removing cores from a preconfigured GIC on page 361

Table C-7: Differences between issue 0100-03 and issue 0100-07

Change	Location
Updated the description about reassigning SPIs with GICD_IVIEWRn.	4.5 Multi view on page 63
Added information about the excessive assertion of the pmu_int signal.	Overflow interrupt on page 96
Changed GICT_ERR1MISC0 to GICT_ERR<n>MISC0.	4.17.4.2 SPI RAM error records 1-2 on page 105
Updated the SPF bit and GICTNS bit descriptions.	5.2.6 GICD_SAC, Secure Access Control register on page 149
Updated the CC_SHARED bit description.	5.2.12 GICD_CCCTLR, Cross-Chip Control Register on page 157
Updated the SPI_BLOCK_MIN and SPI_BLOCKS descriptions.	5.2.15 GICD_CHIPR<n>, Chip Registers on page 160
Updated the ACRC bit description.	Table 5-32: GICD_CFGID bit assignments on page 175
Added the GSPV_CGO bit.	5.4.7 GICR_FCTLR, Function Control Register on page 198
Added two usage constraints.	5.4.10 GICR_VIEWR, View Register on page 202
Added a usage constraint.	5.4.11 GICR_FLUSHR, Flush Register on page 203
Updated the usage constraint.	5.4.12 GICR_MPIDR, MPIDR Register on page 205
Corrected the width of the PPIs_per_Processor field.	5.5.10 GICR_CFGID1, Configuration ID1 Register on page 217
Updated the OF bit description.	5.10.3 GICT_ERR<n>STATUS, Error Record Primary Status Register on page 253
For the SMID field, added information about setting EN and SMID=255.	5.12.6 FMU_SMEN, Safety Mechanism Enable register on page 303
Corrected the bit names and description for the AXI5-Stream cross-chip protection.	• 5.12.10 FMU_SMWDATA, Safety Mechanism Write Data register on page 309 • 5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
Added information about page 1 timeouts for ACE5-Lite cross-chip protection.	• 5.12.10 FMU_SMWDATA, Safety Mechanism Write Data register on page 309 • 5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
Corrected the error_INTID field width and description for page 2 interrupt protection.	5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
Added a restriction for the SMID field when BLKID_ERR == 1 or BLKID_PWROFF == 1.	5.12.13 FMU_STATUS, FMU Status Register on page 321
Added ACE5-Lite to the SM_AXITCRC_GICD_ICDR description.	6.1.6 Protection mechanism IDs on page 336
Updated the GSPV error recovery description.	GSPV error recovery on page 349
Clarified the behavior of successive writes to the FMU_KEY register.	6.1.8 Lock and key mechanism on page 353
Added information about interrupt protection BIST.	6.9 Interrupt protection BIST and the DetectionPaused bits on page 360
Added a limitation for the FMU.	A.1 Removing cores from a preconfigured GIC on page 361

Table C-8: Differences between issue 0200-05 and issue 0200-08

Change	Location
Added information about the FMU.	3.1 Distributor (GICD) on page 26
Corrected the MTE_Support information for ACE5-Lite manager interfaces.	3.1.3.1 AMBA bus properties, GICD manager interface on page 33
Added an option for the number of real-time GCIs.	3.1.6 Distributor configuration on page 35
Updated both data bus widths for a real-time GCI.	3.2.5 GCI configuration on page 40
Added the spi_col_id[4:0] signal to the SPI Collator figure.	3.5 SPI Collator on page 50
Corrected the names of the `SPI_PROT_RESET_DISABLED` and `SPI_PROT_RESET_PERMONLY` build-time options.	3.5.6 SPI Collator configuration on page 52
Clarified the register programming order when GICD_CFGID.RDC == 1. Updated the description about reassigning SPIs with GICD_IVIEWRn.	4.5 Multi view on page 63
Changed SPI IDs to INTIDs, in the NumSPIS description.	5.3.1 GICM_TYPER, Message-based Type Register on page 186
Corrected the GICR_FCTLR reset value for a real-time GCI.	5.4 Redistributor registers for control and physical LPIs summary on page 188
Updated the ECP bit description.	5.4.7 GICR_FCTLR, Function Control Register on page 198
Added two usage constraints.	5.4.10 GICR_VIEWR, View Register on page 202
Added a usage constraint.	5.4.11 GICR_FLUSHR, Flush Register on page 203
Updated the usage constraint.	5.4.12 GICR_MPIDR, MPIDR Register on page 205
Corrected the width of the PPIs_per_Processor field.	5.5.10 GICR_CFGID1, Configuration ID1 Register on page 217
Added the VID and ChipOffset fields.	5.6.3 GICR_VINVCHIPR, vPE Invalidate Chip Register on page 223
Added some Q-Channel information to events `0x94` and `0x95`.	Table 5-115: GICP_EVTYPERn bit descriptions on page 272
Added a usage constraint about clearing the V bit.	5.12.3 FMU_ERR<n>STATUS, Error Record <n> Primary Status register on page 297
Corrected the error_INTID field width and description for page 2 interrupt protection.	5.12.12 FMU_SMRDATA, Safety Mechanism Read Data register on page 316
Added Q-Channel information.	6.1.2 FMU Q-Channel on page 334
Updated the GSPV error recovery description.	GSPV error recovery on page 349
Added a recommendation to enable the block error signals in all blocks.	6.1.6.4 Enabling or disabling both error signals on a block on page 351
Added information about interrupt protection BIST.	6.9 Interrupt protection BIST and the DetectionPaused bits on page 360

Table C-9: Differences between issue 0100-07 and issue 0100-09

Change	Location
Changed 0001-07 to 0100-07 in the Document history table.	Release Information on page 2
Added information about the FMU.	3.1 Distributor (GICD) on page 26

Change	Location
Corrected the MTE_Support information for ACE5-Lite manager interfaces.	3.1.3.1 AMBA bus properties, GICD manager interface on page 33
Added the spi_col_id[4:0] signal to the SPI Collator figure.	3.5 SPI Collator on page 50
Corrected the names of the `SPI_PROT_RESET_DISABLED` and `SPI_PROT_RESET_PERMONLY` build-time options.	3.5.6 SPI Collator configuration on page 52
Updated the GICD_IVIEWR[1:0] and GICD_ICVERRR0 descriptions because those registers are Reserved.	5.2 Distributor registers (GICD/GICDA) summary on page 135
Changed SPI IDs to INTIDs, in the NumSPIS description.	5.3.1 GICM_TYPER, Message-based Type Register on page 186
Added the VID and ChipOffset fields.	5.6.3 GICR_VINVCHIPR, vPE Invalidate Chip Register on page 223
Added a usage constraint about clearing the V bit.	5.12.3 FMU_ERR<n>STATUS, Error Record <n> Primary Status register on page 297
Corrected the PAGEID field description.	• 5.12.9 FMU_SMWR, Safety Mechanism Page Write Register on page 308 • 5.12.11 FMU_SMRD, Safety Mechanism Page Read Register on page 314
Added Q-Channel information.	6.1.2 FMU Q-Channel on page 334
Added a recommendation to enable the block error signals in all blocks.	6.1.6.4 Enabling or disabling both error signals on a block on page 351

Table C-10: Differences between issue 0200-08 and issue 0200-10

Change	Location
Changed 0001-07 to 0100-07 in the Document history table	Release Information on page 2