Arm® CoreLink™ GIC-700T Generic Interrupt Controller

Revision: r0p0

Technical Reference Manual

Non-Confidential

Issue 01
110018_0000_01_en

Arm® CoreLink™ GIC-700T Generic Interrupt Controller

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	30 October 2024	Non-Confidential	First early access release for r0p0

Proprietary Notice

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-700T.

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 <u>underline</u>	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.

Caution

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

Warning

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

Danger

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

Note

This information is important and needs your attention.

Tip

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

Remember

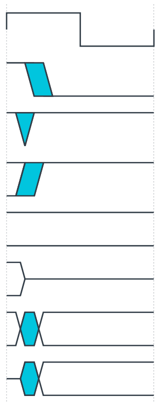
This information reminds you of something important relating to the current content.

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 0615	Confidential
Arm® CoreLink™ GIC-700T Generic Interrupt Controller Configuration and Integration Manual	110019	Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® AXI Protocol Specification	IHI 0022J	Non-Confidential
AMBA® AXI-Stream Protocol Specification	IHI 0051B	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
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
Learn the architecture - Generic Interrupt Controller v3 and v4, LPIs	102923	Non-Confidential
Learn the architecture - Generic Interrupt Controller v3 and v4, Overview	198123	Non-Confidential
Learn the architecture - Generic Interrupt Controller v3 and v4, Virtualization	107627	Non-Confidential

Non-Arm resources	Document ID	Organization
Standard Manufacturer's Identification Code	JEP106BK	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-700T

The GIC-700T is a generic interrupt controller that handles interrupts from peripherals to the cores, and interrupts between cores. The GIC supports a single cluster only, and the cluster can have up to 4 cores.

The GIC-700T supports the GICv3, GICv3.1, GICv3.3, GICv4.1, and GICv4.2 architectures, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

2.1 Component overview

The GIC-700T comprises several significant blocks that work in combination to create a single architecturally compliant GICv3, v3.1, v3.3, v4.1, and v4.2 implementation within the system.

The GIC-700T 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) and also *Locality-specific Peripheral Interrupts* (LPIs).
It is responsible for the entire GIC programmers model, including the GITS_TRANSLATER register.
In configurations that support GICv4.1 and GICv4.2, the Distributor also manages vSGIs and the management of vPEs.

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 the cores. The GCI can scale from 1-4 cores and is best placed next to the processors that it is servicing to reduce wiring to the cores.
The GCI is also referred to as a Redistributor.

The GIC architecture specifies a Redistributor address space containing 2 pages for each core for GICv3 and 4 pages for each core for GICv4.1 and v4.2. The SGI page functionality is contained in the GIC-700T Redistributor. However, the Distributor contains the other pages for all cores on the chip.

The GIC-700T supports powering down the GCI 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-700T supports 1 ITS block only.

For more information about the ITS, see the *Learn the architecture - Generic Interrupt Controller v3 and v4, LPIs*.

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-700T supports up to 1984 SPIs that are spread across the system. The SPI Collator enables 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 can be spread across 32 SPI Collators, with a maximum of 1024 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.

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-700T Generic Interrupt Controller Configuration and Integration Manual* and the interconnect documentation.

The GIC-700T implements version 3, v3.1, v3.3, v4.1, and v4.2 of the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*. To use GIC-700T with a core, the core must:

- Implement any of the Armv8.x-A or Armv9.x-A architectures and support the GIC Stream protocol.
- Support the extended range of GICv3.1 interrupts, when GIC-700T is configured and programmed to use >960 SPIs or >16 PPIs for each core.
- Support GICv4.1 and GICv4.2, when GIC-700T is configured and programmed to use these GICv4 features.

2.2 Compliance

The GIC-700T interfaces are compliant with Arm specifications and protocols.

The GIC-700T 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, 3.3, 4.1, and 4.2 of the Arm GIC architecture specification. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.
- 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-700T provides interrupt services and masking, registers and programming, interrupt grouping, security, performance monitoring, and error correction.

Interrupt services and masking

The GIC-700T 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-700T.
    - 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 48 *Private Peripheral Interrupts*) (PPIs) that are independent for each core and can be programmed to support either edge-triggered or level-sensitive interrupts.
    - Up to 16 physical *Software Generated Interrupts* (SGIs) for each core, which the core generates through its GIC CPU interface.
    - Direct injection of up to 16 virtual SGIs for each vPE, when the GIC is configured to support GICv4.1.
- A single *Interrupt Translation Service* (ITS) module that provides device isolation and ID translation for message-based interrupts and enables virtual machines to program devices directly.
- Interrupt masking and prioritization with 32 priority levels, 5 bits for each interrupt.

Registers and programming

The GIC-700T 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 for programming of all registers, which includes the GITS_TRANSLATER register.

Security

The GIC-700T provides the following security features:

- A global *Disable Security* signal. The gicd_ctlr_ds signal enables support for systems without security support.
- The following interrupt groups allow interrupts to target different Exception levels:
    - Group 0
    - Non-secure Group 1
    - Secure Group 1

  See 4.2 Interrupt groups and security on page 41 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-700T provides *Performance Monitoring Unit* (PMU) counters with snapshot functionality.

Error correction and containment

The GIC-700T 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.

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.

2.4 Test features

The GIC-700T provides *Design for Test* (DFT) signals for test mode.

Related information

Common control signals on page 238

2.5 Product documentation

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

For relevant protocol and architectural information that relates to this product, see 1.4 Useful resources on page 14.

The GIC-700T documentation is as follows:

Technical Reference Manual

The TRM describes the functionality and the effects of functional options on the behavior of the GIC-700T. 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-700T, contact:

- The implementer to determine:

    - The build configuration of the implementation
    - What integration, if any, was performed before implementing the GIC-700T
- 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-700T into an SoC
- How to implement the GIC-700T 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.

3. Components in GIC-700T

The GIC-700T contains several major components that use an internal GIC interconnect to route the AXI5-Stream interfaces between the different components.

The components are:

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

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

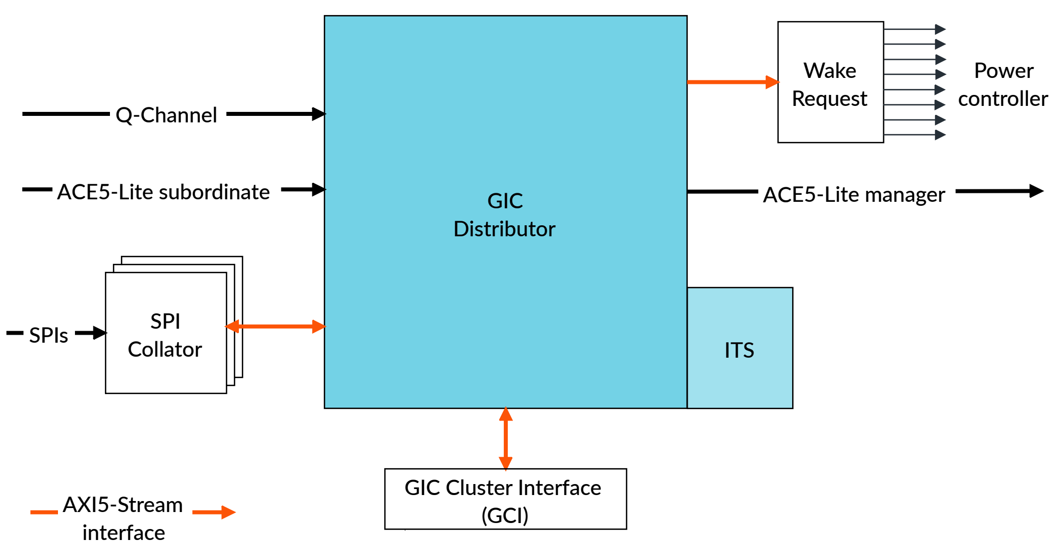
3.1 Distributor (GICD)

The Distributor is the main communication point between all GIC-700T blocks. It performs SPI management and LPI caching, and all communications with other blocks.

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.

Figure 3-1: GIC-700T Distributor



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, see 4.8 SPIs on page 52.
- Caching and forwarding of LPIs, see 4.10 LPIs on page 61.
- SGI routing and forwarding, see 4.6 SGIs on page 49.
- 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, including the GITS_TRANSLATER.
- Power control of cores and Redistributors.

3.1.1 Distributor AXI5-Stream interfaces

The GIC-700T uses AXI5-Stream interfaces to communicate between blocks. These interfaces are internal and are not exposed to the system.

3.1.2 Distributor ACE5-Lite subordinate interface

The AMBA® ACE5-Lite subordinate port on the GIC-700T Distributor provides access to the entire register map including the GITS_TRANSLATER register. The interface supports 64-bit, 128-bit, 256-bit, or 512-bit data widths.

The GIC-700T only accepts single beat accesses of the sizes for each register that are shown in the programmers model, see 5. Programmers model for GIC-700T on page 101.

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 31.

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

Table 3-1: 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-700T 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-700T 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-700T 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-700T 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-700T 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 101.

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 3-2: Subordinate response signaling for ECC detection errors

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

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

The following table lists the ACE5-Lite subordinate interface properties.

Table 3-3: GICD 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	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
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

AMBA property	Subordinate interface	ACE5-Lite issue
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.

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

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

Table 3-4: 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	`mpfa_count` of the ITS. `mpfa_count` is a configuration parameter.	1
256-bit read of ITS command queue	1	0
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

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 3-5: 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	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
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	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-Channel

There is a single Q-Channel for clock gating the GIC-700T Distributor. The Q-Channel interface denies access when the Distributor is busy processing interrupts.

The qreqn* signals are synchronized internally, and can be driven asynchronously. See B.2 Power control signals on page 239.

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 configuration

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

Table 3-6: Configurable options for the Distributor

Feature	Range of options
Affinity0 width	0-4
Affinity1 width	0-8
Affinity2 width	0-8
Affinity3 width	0
Number of *GIC Cluster Interfaces* (GCIs)	1
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	0-1
GICv4.1 support	True, False
GICv4.2 support	True, False
Number of vPEs supported, $2^{\text{<value>}}$	2-14
Number of SPI signals	32-1984, in blocks of 32. The 1984 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.

Feature	Range of options
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.
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. See B.7 RAM I/O signals on page 248.
Remove support for 1 of N SPIs	True, False

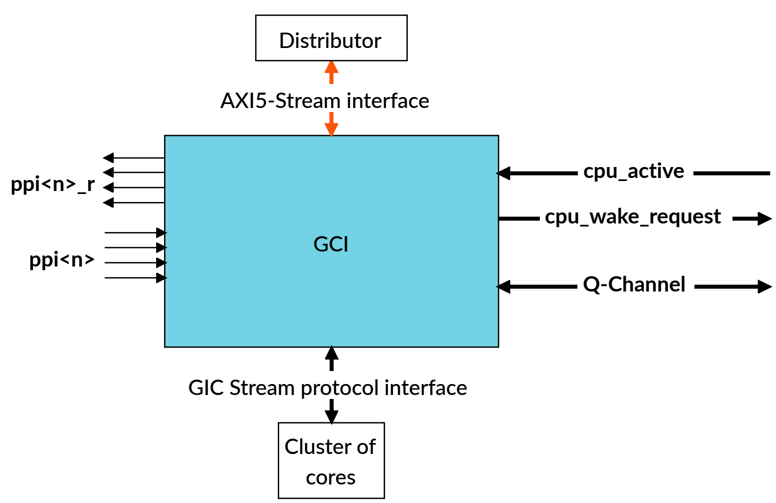
For more information, see the *Arm® CoreLink™ GIC-700T Generic Interrupt Controller Configuration and Integration Manual*.

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.

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.
- Maintaining the GIC Stream protocol and communicating with the cluster.

The GIC-700T supports one GCI. The GCI supports a single cluster of up to 4 cores.

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

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 38.

Related information

PPIs on page 50

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 internal and is not exposed to the system.

3.2.2 GCI GIC Stream Protocol interface

The GIC-700T 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.

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 3-7: 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.

Signal	Description
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.

Both the iritdest and icctid signals can support 4 cores that use packed binary encoding, as opposed to one-hot encoding.

3.2.3 GCI PPI signals

GIC-700T 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 the 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:

- `GIC700T_<usrcfg>_PPI0_<cpu_number>_<ppi_number>_INV`
- `GIC700T_<usrcfg>_PPI0_<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 `GIC700T_<usrcfg>_PPI0_<cpu_number>_<ppi_number>_INV` parameter also inverts the ppi<n>_r signal.

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 51
SPI Collator wires on page 35

3.2.4 GCI configuration

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

Table 3-8: Configurable options for the GCI

Feature	Range of options
The number of cores that attach to this GCI.	1-4
The number of PPIs for each core. To support more than 16 PPIs, the core must support the GICv3.1 extensions.	16, 32, 48

Feature	Range of options
ECC support for the RAM. See 4.14 Reliability, Accessibility, and Serviceability on page 71 for more information.	True
Data bus width for the GCI processor AXI5-Stream interface.	16, 32
AXI5-Stream data bus width.	16, 32, 64

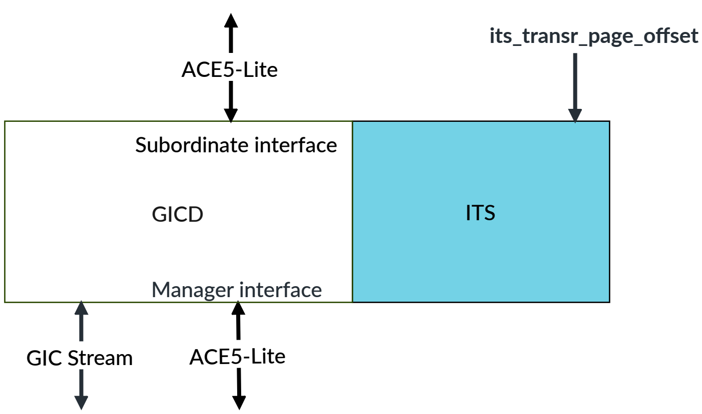
For more information, see the *Arm® CoreLink™ GIC-700T Generic Interrupt Controller Configuration and Integration Manual*.

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 single ITS block that the GIC supports.

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).

See 4.9 ITS on page 57 for more information.

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

The GIC-700T supports 2 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 2 modes cannot be mixed on the ITS. The DeviceID must be transferred using a method that malicious software cannot spoof. The system is responsible for ensuring that the DeviceID reaching the ITS, is not spoofed by rogue software using a<x>user signals or MSI-64 mode. See 3.4 MSI-64 Encapsulator on page 32.

3.3.1 ITS configuration

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

Table 3-9: Configurable options for the ITS

Feature	Range of options
DeviceID width	3-24
EventID width	1-20
CollectionID width	2-14
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 4.9.2 MSI-64 on page 59.	True, False
ECC support for the caches. For more information, see 4.14 Reliability, Accessibility, and Serviceability on page 71.	True, False
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

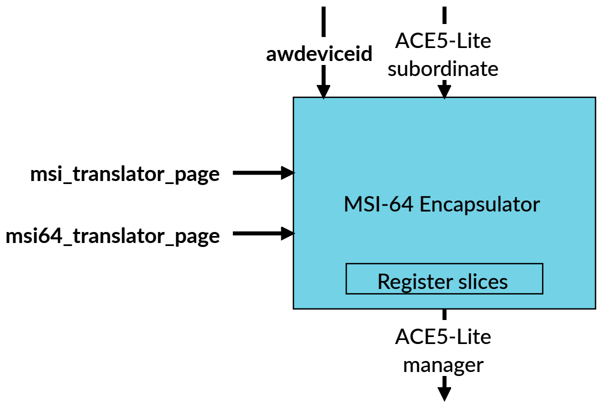
For more information, see the *Arm® CoreLink™ GIC-700T Generic Interrupt Controller Configuration and Integration Manual*.

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.

Figure 3-4: 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 4.9.2 MSI-64 on page 59 for more information.

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 3-10: 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. These signals are described in B.5 ACE5-Lite interface signals on page 242.

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 3-11: 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

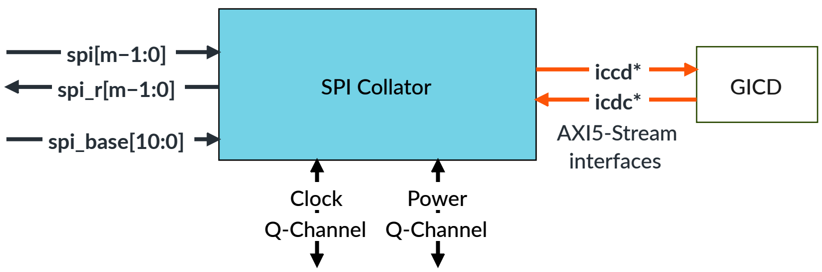
Build-time option	Function	Range of options
REV_REG_TYPE	Register slice type on B and R channels.	**0** None **1** Reverse **2** Forward **3** Full

3.5 SPI Collator

The SPI Collator converts SPI wires into messages to be sent to the Distributor.

The following figure shows an SPI Collator block.

Figure 3-5: 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:

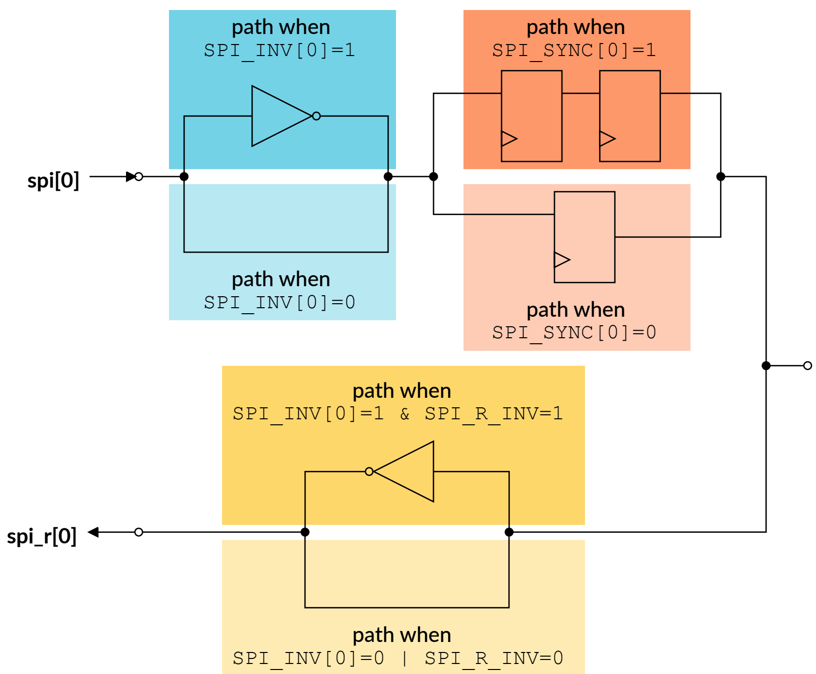
- `SPI_INV[n] == 1` indicates that the inverter is enabled.
- `SPI_SYNC[n] == 1` indicates that the synchronizer is enabled.

- `[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 3-6: 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.

Related information

Miscellaneous signals on page 246

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 4.12 Power management on page 66 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 3-12: Configurable options for an SPI Collator

Feature	Range of options
The number of SPI wires. The total number of SPIs on all SPI Collators must be ≤1984.	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 35.	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.

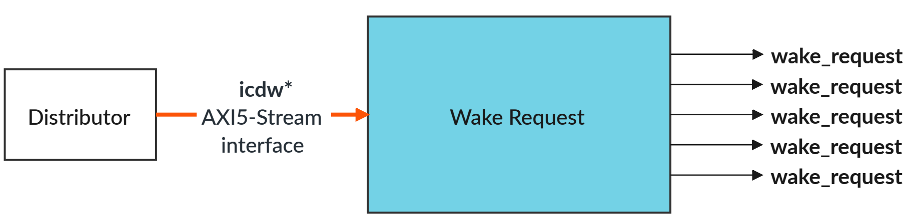
For more information, see the *Arm® CoreLink™ GIC-700T Generic Interrupt Controller Configuration and Integration Manual*.

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.

Figure 3-7: 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-700T 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-700T 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.

See A.1 Removing cores from a preconfigured GIC on page 232 for more information.

Related information

Power control signals on page 239

3.6.1 Wake Request configuration

The Wake Request block has a single configuration option.

Table 3-13: Configurable options for Wake Request

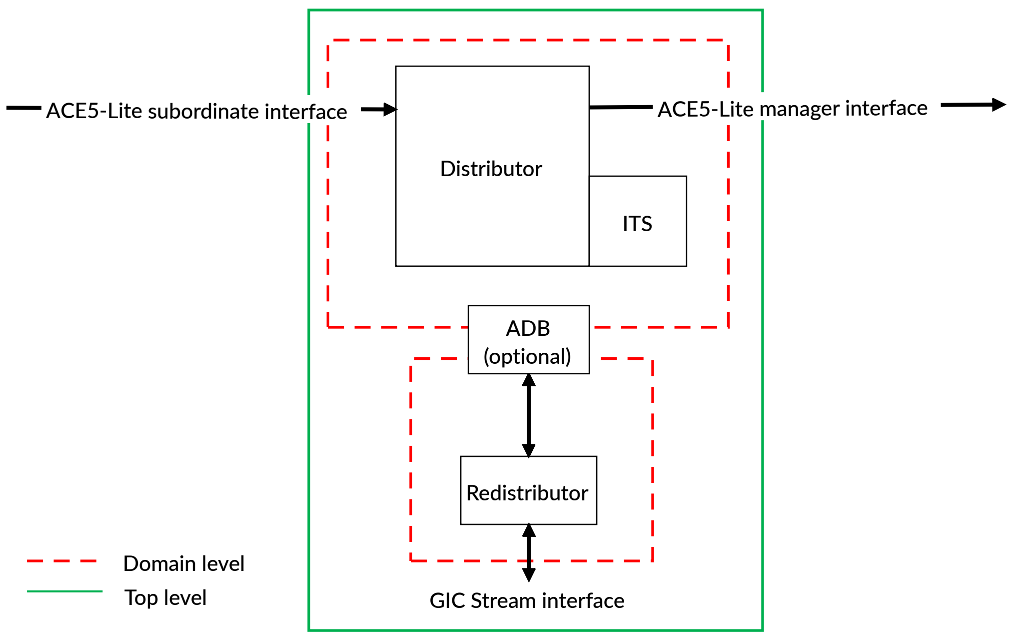
Feature	Range of options
Compress the width of the wake_request[<cpus> − 1:0] signal	0, 1

3.7 Hierarchy

The hierarchy of the GIC components is fixed because the `structure` configuration parameter is always set to `full`. Therefore, all the GIC blocks are stitched together to create a single top-level GIC-700T file, `gic700t_<usrcfg>.v`.

The following figure shows the top-level options.

Figure 3-8: GIC top-level structure options



4. Operation of GIC-700T

This chapter provides an operational description of the GIC-700T product.

4.1 Interrupt types

The GIC-700T manages SPIs, SGIs, PPIs, and LPIs. When GICv4.1 is enabled, SGIs and LPIs can be injected into *virtual Processing Elements* (vPEs).

4.2 Interrupt groups and security

The GIC-700T 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-700T 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-700T security as follows:

gicd_ctlr_ds is LOW

Security enabled

gicd_ctlr_ds is HIGH

Security disabled

Setting the gicd_ctlr_ds tie-off signal HIGH removes the security support of the GIC-700T. 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 - Generic Interrupt Controller v3 and v4, Overview*.

4.3 Affinity routing and assignment

The GIC-700T 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 3 levels of affinity, with an implicit affinity level 3 value of 0.
- For 64-bit Armv8 and Armv9 processors, the MPIDR defines 4 levels of affinity.

The GIC-700T 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 Aff`n` 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 and Armv9-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-700T allows fully flexible allocation of MPIDR. However, it has 2 built-in default assignments that are based on the `aff0_thread` configuration parameter:

`aff0_thread` == 1

The 4 fields map to 0.<cluster>.<core>.<thread>

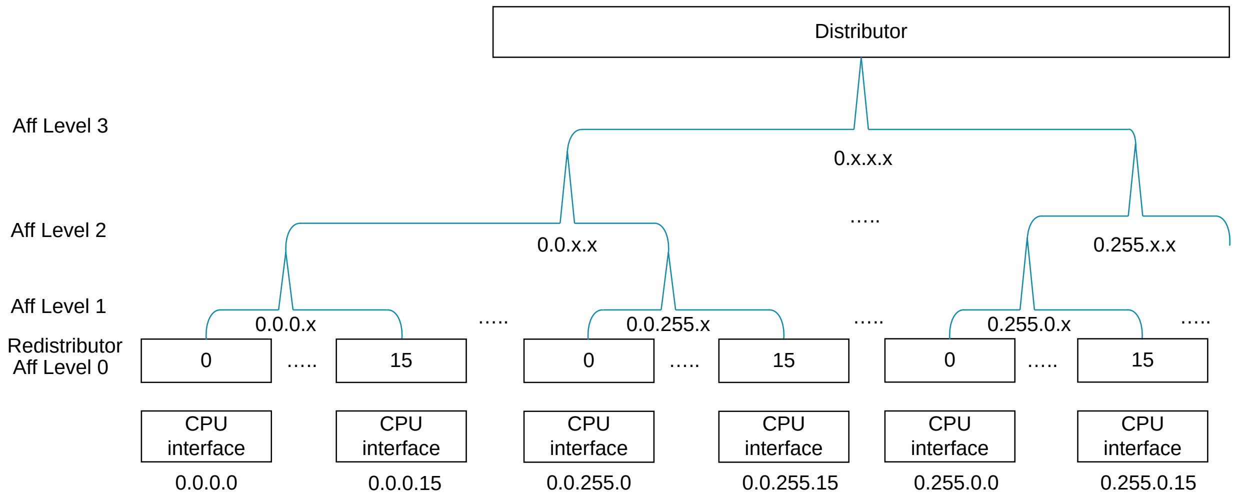
`aff0_thread` == 0

The 4 fields map to 0.0.<cluster>.<core>

See the *Arm® CoreLink™ GIC-700T 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-700T 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. GIC-700T supports only 4 cores.

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 232 for more information.

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

4.4 RAMs and ECC

The GIC-700T 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 can be protected from errors using an ECC with *Single Error Correction and Double Error Detection* (SECDED). See the *Arm® CoreLink™ GIC-700T Generic Interrupt Controller Configuration and Integration Manual* for information about the ECC configuration parameters.

If single or double errors are detected, they are reported in the software visible error records, see 4.14 Reliability, Accessibility, and Serviceability on page 71 for more information.

4.4.1 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

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

GITS_C_ERRINSR, Error Insertion Collection cache register on page 189
GITS_D_ERRINSR, Error Insertion Device cache register on page 185
GITS_V_ERRINSR, Error Insertion Event cache register on page 187

4.4.2 Scrub

The GIC-700T 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.

4.5 Direct injection

The GIC-700T 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-700T 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.
- The valid data fields of all `VMAPP` commands for each vPE 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.

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 the GITS_BASER2 register, 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.

4.5.1 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-700T has the following doorbell characteristics:

- Doorbell IDs must be unique and not mapped to any DeviceID and EventID on the ITS.
- GIC-700T 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 4.5.2 Residency and VMOVP on page 47.
- 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.

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.

4.5.2 Residency and VMOVP

Software freely moves vPEs around between PEs using the ITS `VMOVP` command.

Doorbells are delivered to the PE that the vPE is mapped to. For more information on doorbells, see 4.5.1 Doorbells on page 46. 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 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.
- Make a vPE resident when not mapped with a prior `VMAPP` command.
- 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 2 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.

4.5.3 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:

1. Issue `VMAP(V0A0)` commands on the ITS.
2. Issue `VMAPP(V1A1)` on the ITS with a vPT_size of at least as large as the original.
3. Issue `VMAPP(V0A1)` on the 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.

4.6 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.

4.6.1 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 153.

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.

4.6.2 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 4.14.4.3 SGI RAM error records 3-4 on page 79.

4.6.3 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 50 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 50.

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 79

4.7 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-700T provides an option, through parameters, to include a synchronizer or inverter, or both, on each PPI interrupt signal. See 3.2.3 GCI PPI signals on page 30 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.

4.7.1 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 41 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 - Generic Interrupt Controller v3 and v4, Overview* and the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

4.7.2 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 153

4.7.3 PPI direct injection

The GIC-700T cannot directly inject PPIs into vPEs.

4.7.4 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 can be 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 4.14.4.5 PPI RAM error records 7-8 on page 80.

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.

4.8 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. 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-700T 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 are generated either by wire inputs or by writes to the ACE5-Lite subordinate programming interface. The GIC-700T can support up to 1984 SPIs corresponding to the spi input signals on the SPI Collators. Each SPI Collator has a limit of 1024 signals. The number of SPIs available depends on the implemented configuration. The first SPI has an ID number of 32. The permitted ID values are in steps of 32, in the following ranges:

- ID32-ID991
- ID4096-ID5119

During configuration of the GIC, you can allocate some or all SPIs to be message-based or you can set all SPIs to be a physical 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.

You can configure whether each SPI is triggered on a rising edge or is active-HIGH level-sensitive. The GIC-700T provides a build-time option, to include one or both of a synchronizer or inverter for each spi signal.

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 35 for more information.

SPIs are programmed through the GICD register address space 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*.

4.8.1 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 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 35 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 - Generic Interrupt Controller v3 and v4, Overview* and the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

The GIC-700T provides optional synchronizers on every interrupt wire input. The GIC also provides return signals, spi_r, to support interrupt pulse extension when sending edge-triggered interrupts across domain boundaries, see 3.5.2 SPI Collator wires on page 35.

4.8.2 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 104

4.8.3 SPI routing and 1 of N selection

If GICD_TYPER.No1N==0, then the GIC-700T supports 1 of N selection of SPI interrupts. You can program an SPI to target several cores, and the GIC-700T 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-700T only sends an SPI to cores that are powered up and have the relevant interrupt group enabled. The GIC-700T prioritizes cores that are considered active, but if there are no active cores, it selects inactive cores.

The selections that the GIC-700T 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-700T 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-700T 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-700T 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.

4.8.4 SPI direct injection

The GIC-700T cannot directly inject SPIs into vPEs.

4.8.5 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 56
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 56 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.

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 56.

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

Related information

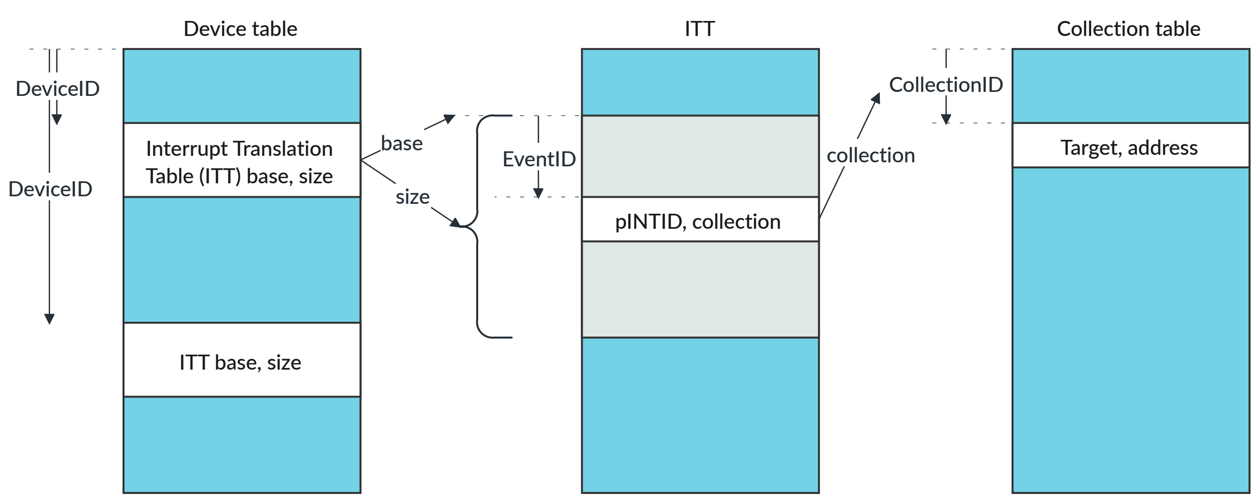
SPI RAM error records 1-2 on page 78

4.9 ITS

The *Interrupt Translation Service* (ITS) is responsible for translating message-based interrupts from peripherals into LPIs or vLPIs. The GIC-700T supports a single ITS.

The ITS 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 4-2: Physical ITS process



To reduce memory traffic and keep interrupt latency to a minimum, GIC-700T has three 2-way set associative caches in the ITS:

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

If ECC protection is not required for a cache, you can remove ECC from each RAM individually. See the *Arm® CoreLink™ GIC-700T Generic Interrupt Controller Configuration and Integration Manual* for more information.

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-700T 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-700T 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-700T.

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

The GIC-700T 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 4.14 Reliability, Accessibility, and Serviceability on page 71.

For information about how to program and use the ITS, see the *Learn the architecture - Generic Interrupt Controller v3 and v4, LPIs*.

4.9.1 ITS cache control, locking, and test

The GIC-700T 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-700T 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.

4.9.2 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-700T 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.

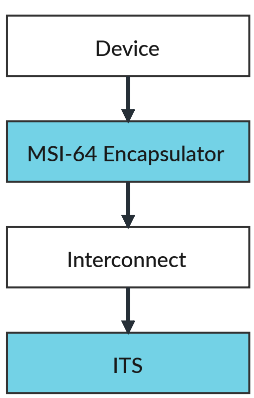
The GIC-700T has the following features to support this:

- 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 32 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 4-3: MSI-64 Encapsulator with DeviceID sent in the data[63:32] bits



4.9.3 ITS commands and errors

The 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
Translation errors on incoming writes to GITS_TRANSLATER	GITS_FCTLR.UEE (Unmapped Error Enable)
Errors during commands	GITS_FCTLR.CEE (Command Error Enable)
Other errors such as memory system, or memory allocation errors	None

See 4.14.4.15 ITS command and translation error records 27 on page 87 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 4.14.3 Error recovery and fault handling interrupts on page 71 for more information.

4.10 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.

4.10.1 LPI programming and generation

Only an ITS can generate an LPI. See *Learn the architecture - Generic Interrupt Controller v3 and v4, LPIs* for more information.

4.10.2 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.

4.10.3 LPI operation

The GIC-700T does not use physical target addresses, so GITS_TYPER.PTA == 0. Therefore, GIC-700T uses the value of GICR_TYPER.ProcessorNumber to route all LPIs and commands to their targets.

4.10.4 LPI caching

If LPI support is configured, the GIC-700T supports a single LPI cache 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-700T 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-700T 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 27

4.10.5 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-700T 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-700T 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-700T attempts to automatically balance the interrupt load to cores that are active but not handling other interrupts.
- The GIC-700T 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-700T, 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 - Generic Interrupt Controller v3 and v4, Overview*.

4.10.6 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.

4.11 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 and Armv9 cores. However, setting the relevant DCC bits converts the GIC-700T to full featured mapping.

Table 4-4: Memory access registers

Access type	Register	Mapping control bit
LPI Property table	GICR_PROPBASER	GICD_FCTLR2.DCC

Access type	Register	Mapping control bit
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`
`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*.

4.11.1 MPAM information

The GIC-700T 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 the 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 ITS.

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

During the GIC configuration process, the maximum supported number of bits of PARTID and PMG is set. The default values are PARTID = 9 and PMG = 1, which aligns with most ACE5-Lite interface implementations. To discover the values that a GIC configuration supports, software can read GICR_MPAMIDR or GITS_MPAMIDR.

4.12 Power management

The GIC-700T 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.

4.12.1 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-700T 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.

4.12.2 Processor core power management

The GIC architecture defines the programming sequence to safely power down a core that connects to the GIC-700T.

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-700T 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 38.

You must not set the GICR_WAKER.ProcessorSleep bit to 1, unless the core enters a power state where the GIC-700T 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.

4.12.3 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*

If the spi_ram_retained signal is HIGH when GIC-700T exits reset, then the GICD trusts the data in the `u_spi_ram0` and `u_spi_ram1` RAM instances.

The GIC-700T 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. See B.7 RAM I/O signals on page 248.

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.

4.13 Performance Monitoring Unit

The GIC-700T contains a PMU for counting the main GIC events from the Distributor and the ITS.

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 5-101: GICP_EVTYPERn.EVENT field encoding on page 215.

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.

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.

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, 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.

4.14 Reliability, Accessibility, and Serviceability

The GIC-700T 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.

4.14.1 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.

4.14.2 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.

4.14.3 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 or out of range, 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.

4.14.4 Error handling records

The GIC-700T has several error records. The range of error handling records that are available depends on the configuration of the GIC-700T.

The following table lists the GIC-700T 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 74
1	Correctable SPI RAM errors	CE	Table 4-8: SPI RAM errors, records 1-2 on page 79. 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 80. 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 80. 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 81. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
8	Uncorrectable PPI RAM errors	UER	

Record	Description	Type	Description, events, and recovery sequences
9	Correctable LPI RAM errors	CE	Table 4-12: LPI RAM errors, records 9-10 on page 82. 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	4.14.4.7 PTS RAM error records 11-12 on page 82. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
12	Uncorrectable PTS RAM errors	UER	
13	Correctable TGT_LPI RAM errors	CE	4.14.4.8 TGT_LPI RAM error records 13-14 on page 82. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
14	Uncorrectable TGT_LPI RAM errors	UER	
15	Correctable VICM RAM errors	CE	4.14.4.9 VICM RAM error records 15-16 on page 83. 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	4.14.4.10 VSPA RAM error records 17-18 on page 83. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory.
18	Uncorrectable VSPA RAM errors	UER	
19	Correctable VTGT_VSTR RAM errors	CE	4.14.4.11 VTGT_VSTR RAM error records 19-20 on page 84. 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	4.14.4.12 VTGT_VRES RAM error records 21-22 on page 85. 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	4.14.4.13 VTGT_SRCH RAM error records 23-24 on page 85. 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 86. GICT_ERR<n>STATUS.SERR == 6, data value from associative memory.
26	Uncorrectable error from ITS RAM	UEO	
27	ITS command and translation errors	UER	4.14.4.15 ITS command and translation error records 27 on page 87. GICT_ERR<n>STATUS.SERR == 14, illegal access.

4.14.4.1 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-700T. 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 4.14.3 Error recovery and fault handling interrupts on page 71 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.EnableLPIs 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 235.
`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.EnableLPIs 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 235. Otherwise, repeat the register access using the current global values.
`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.

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
`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.
`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.	`0xF`	ID, bits[9:0]	The system is misconfigured. Check the base tie-offs on the SPI Collators.
`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.
`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.

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
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.	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.
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 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.	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.

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
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.
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). • The GITS_BASER2 register with GITS_CTLR.Enabled == 1.

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.
`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.EnableLPIs is set.

4.14.4.2 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-700T 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 4.14.3 Error recovery and fault handling interrupts on page 71 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 56

4.14.4.3 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 4.14.3 Error recovery and fault handling interrupts on page 71 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 4 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 50

4.14.4.4 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.

4.14.4.5 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 4.7.4 PPI error recovery procedure on page 52 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.

4.14.4.6 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 4.10.6 LPI error recovery procedure on page 64 describes.

4.14.4.7 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.

4.14.4.8 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 4.10.6 LPI error recovery procedure on page 64.

4.14.4.9 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-700T 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^{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]

4.14.4.10 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-700T 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]

4.14.4.11 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-700T 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]

4.14.4.12 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-700T 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]

4.14.4.13 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-700T 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, $2^{\mathtt{vpe\_width}}$ 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:$\log_2 2^{vpe\_width}$ / 128] • Address, bits[$\log_2(2^{vpe\_width}$ / 128) − 1:0]
24 = Uncorrectable	When `vpe_width` ≤ 8: • Address, bit[0] When `vpe_width` > 8: • Address, bits[$\log_2(2^{vpe\_width}$ / 128) − 1:0]

4.14.4.14 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

4.14.4.15 ITS command and translation error records 27

The ITS command and translation error records 27 record uncorrectable command and translation errors from the 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. The GITS_TYPER.DevBits field returns the supported range.	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_TGT_OFF	`0x10923`	1	0	-	Specified targetPE (RDnum) has GICR_CTLR.EnableLPI = 0.	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_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` commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
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 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 by previous commands.	RD that specified collection ICID 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

`MOVALL` commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
MOVALL_SRC_TGT_OOR	`0x10E20`	1	0	-	Specified RDbase1 references a nonexistent target. 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_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

`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 commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
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_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 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_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.	{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_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 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.	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_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.	{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_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 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` commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
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
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_VPE_LOST	`0x12D33`	1	0	-	The system has lost the specified vPE. The causes of this issue can be 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` commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**Data (data is only for GITS_TRANSLATER write interrupts not for `INT` commands)**
INT_TGT_OFF	`0x10323`	1	0	-	`INT` received for a target with GICR_CTLR.EnableLPIs disabled. Software must either enable LPI or correct mappings. Target is reported in GICT_ERR<n>MISC1.	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.	{vIntID[15:0], vPEID[`VPE_WIDTH`-1:0]}
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_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 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.	RDbase from command
VMAPP_ENLPI_OFF	`0x12924`	1	0	-	Specified Target (RDnum) has GICR_CTLR.EnableLPI = 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_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 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.	{DoorbellID[15:0], vPEID[`VPE_WIDTH`-1:0]}
VMAPP_VPROP_V	`0x12936`	1	0	-	Specified Target (RDnum) does not have GICR_VPROPBASER.Valid set.	{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.	{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` commands						
Error mnemonic	**Encoding**	**IERR**	**Stall**	**Mask**	**Description**	**MISC1 data**
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.	{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_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_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_VPE_LOST	`0x12233`	1	0	-	Specified vPEID is mapped to a vPEID that the system has lost. The causes of this issue can be 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]}

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_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_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 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_VPE_LOST	0x12143	1	0	-	The system has lost the specified destination vPEID. The causes of this issue can be data corruption, or conflicting programming such as illegal VMAPP sequences.	{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]}

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_VPE_OOR	0x12E31	1	0	-	Sent INVDB command for a vPEID that is outside the GITS_BASER2 and GICR_VPROPBASER configured range.	vPEID
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_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.	{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 4.9.3 ITS commands and errors on page 60.	[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 2-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 2-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

4.14.4.16 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 50
PPI error recovery procedure on page 52
SPI error recovery procedure on page 56
LPI error recovery procedure on page 64

4.14.5 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 4-23: 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

5. Programmers model for GIC-700T

All the GIC-700T 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-700T registers:

- The GIC-700T implements only memory-mapped registers.
- The GIC-700T 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-700T 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-700T guarantees single-copy atomicity for doubleword accesses.

- The GIC-700T supports data only in little-endian format.
- The access types for the GIC-700T 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-700T address map has multiple pages. The number of pages and the address aliasing depends on the GIC configuration.

The following table shows the register map pages.

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	4	GITS	ITS address page.
5	5	GITS (translate)	ITS translation page
-	6	GITS (vSGI)	ITS vSGI page
-	7	Reserved	Reserved
6 + 2×RDnum	8 + 4×RDnum	GICR (LPI)	GICR LPI registers.
7 + 2×RDnum	9 + 4×RDnum	GICR (SGI)	GICR PPI + SGI registers. RDnum is the serial number of each "internal Redistributor", which is from 0 to RDcount−1.
-	10 + 4×RDnum	GICR (vLPI)	GICR vLPI registers
-	11 + 4×RDnum	Reserved	Reserved
8 + 2×RDcount	12 + 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*.

Page offset

The ACE5-Lite address bits[`x`:16] control which GIC register page is accessed. The value of `x` depends on the `axis_addr_width` GICD configuration parameter.

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`.

5.1.1 Discovery

We recommend that the operating system is provided with pointers to the start of the Distributor, the ITS, and the first Redistributor page.

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-700T, 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-700T 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.
- 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-700T 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.

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

5.1.2 GIC-700T register access and banking

The GIC-700T 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-700T 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.

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	0x080nn43B 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
0x0028-0x002C	-	-	-	-	Reserved	-
0x0030	GICD_FCTLR2	RW	0x0	32	Function Control Register 2	No
0x0034	GICD_UTILR	RW	0x0	32	Utilization Register	No

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0038-0x003C	-	-	-	-	Reserved	-
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	-
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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
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	-
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	-

Offset	Name	Type	Reset	Width	Description	Architecture defined?
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 - Generic Interrupt Controller v3 and v4, Overview*. 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
0xA000-0xC7FC	-	-	-	-	Reserved	-
0xC800	GICD_RDOFFR	RW	0x0	64	Redistributor Off Register. Only present when GICD_CFGID.RDC == 1.	No
0xC808-0xD014	-	-	-	-	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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
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
0xF008-0xFFCC	-	-	-	-	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

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.

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 104 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

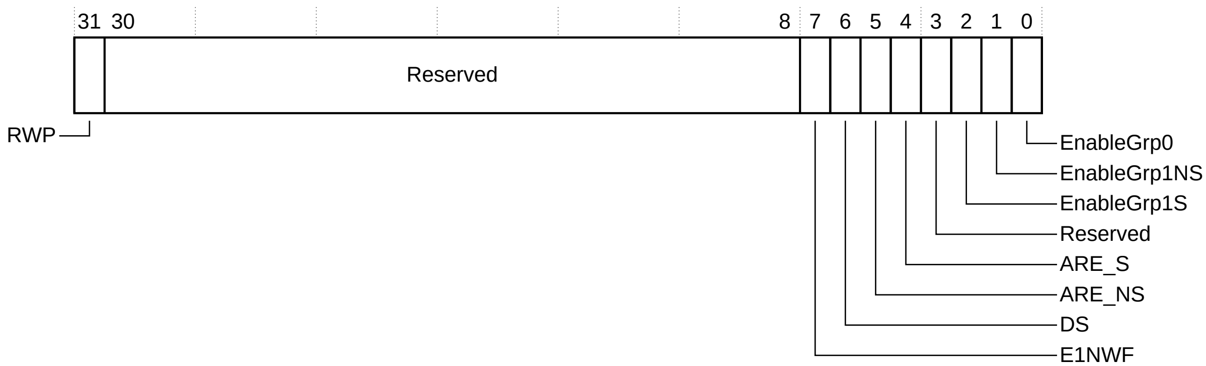


Table 5-3: GICD_CTLR bit descriptions

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 4.2 Interrupt groups and security on page 41 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
[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-700T. 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.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit

Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 104 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

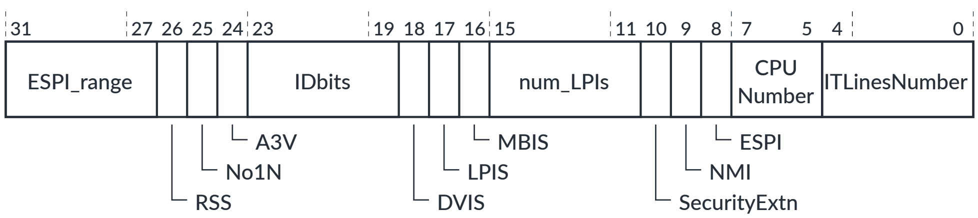


Table 5-4: GICD_TYPER bit assignments

Bits	Name	Description
[31:27]	ESPI_Range	Returns the number of extended SPIs that GIC-700T 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.
[25]	No1N	1 of N SPI: **0** The GIC-700T supports 1 of N SPI interrupts. This value occurs when `spi_1ofn_support` == 1. **1** The GIC-700T does not support 1 of N SPI interrupts. This value occurs when `spi_1ofn_support` == 0.
[24]	A3V	Affinity level 3 values: **0** The GIC-700T Distributor supports only zero values of affinity level 3.
[23:19]	IDbits	Interrupt identifier bits: **`0b01111`** The GIC-700T supports 16 interrupt identifier bits.

Bits	Name	Description
[18]	DVIS	Direct virtual LPI injection support: **0** The GIC-700T does not support direct virtual LPI injection. **1** The GIC-700T does support direct virtual LPI injection. See the *Learn the architecture - 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-700T 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-700T supports only a single Security state. **1** gicd_ctlr_ds signal was LOW during reset, so the GIC-700T 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-700T 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 104 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

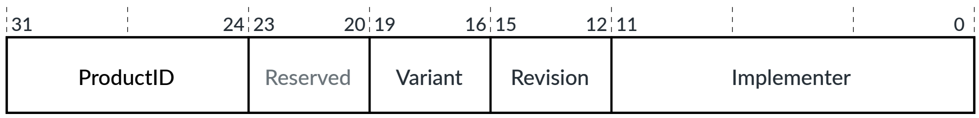


Table 5-5: GICD_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **`0x08`** GIC-700T
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: **`0x0`** r0
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: **`0x0`** p0
[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-700T 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 104 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

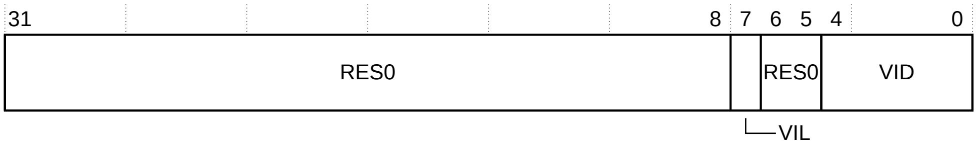


Table 5-6: GICD_TYPER2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, **RES0**.
[7]	VIL	Returns the number of bits that GIC-700T can use for a vPEID: **0** GIC-700T supports 16 bits of vPEID. **1** GIC-700T supports GICD_TYPER2.VID + 1 bit of vPEID. If GICD_TYPER.DVIS == 0, then this bit returns zero.
[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.

5.2.5 GICD_FCTLR, Function Control Register

This register controls non-architectural functionality such as the scrubbing of all RAMs in 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 104 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

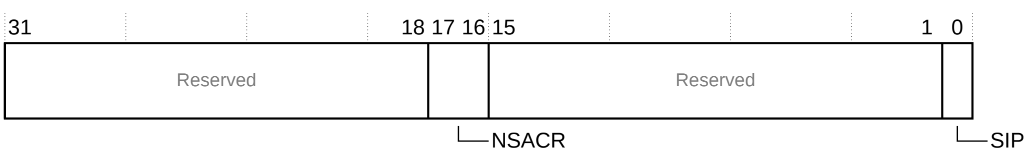


Table 5-7: GICD_FCTLR bit assignments

Bits	Name	Description
[31:18]	-	Reserved, RES0
[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

The NASCR field is accessible only by Secure accesses.

5.2.6 GICD_SAC, Secure Access Control register

This register allows Secure software to grant Non-secure software with access to some GIC-700T Secure features. It also controls whether Secure PMU events are visible to Non-secure software.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit

Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 104 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 5-8: GICD_SAC bit assignments

Bits	Name	Description	Type
[31:4]	-	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.	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.	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.	RW
[0]	-	Reserved, RES0	-

Accessibility

GICD_SAC is accessible only by Secure accesses.

5.2.7 GICD_FCTLR2, Function Control Register 2

This register controls clock gating and other non-architectural controls in 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-7: GICD_FCTLR2 bit assignments

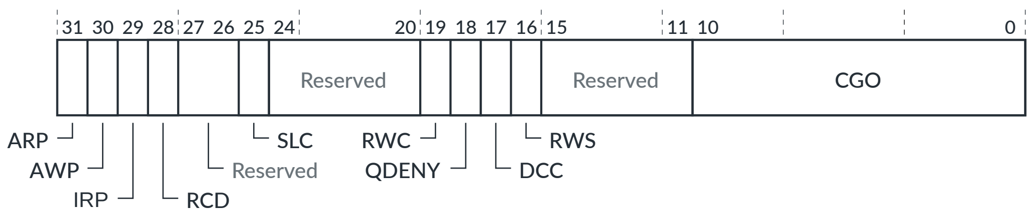


Table 5-9: GICD_FCTLR2 bit assignments

Bits	Name	Description
[31]	ARP	Report read poison if corrupted data from a RAM is read.
[30]	AWP	Report write poison. Reject poisoned writes on the subordinate interface.
[29]	IRP	Ignore read poison from manager.
[28]	RCD	Read chunking disable.
[27:26]	-	Reserved, RES0
[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.
[24:20]	-	Reserved, RES0
[19]	RWC	Residency wait on command. See 4.5.2 Residency and VMOVP on page 47 for more information.
[18]	QDENY	Q-Channel deny. Overrides the Q-Channel logic and forces the Distributor to reject powerdown requests.
[17]	DCC	Do not correct cache. Modifies the a<x>cache output signal from the Distributor. See 4.11 Memory access and attributes on page 64.
[16]	RWS	Residency wait on *Pending Table System* (PTS) RAM search. See 4.5.2 Residency and VMOVP on page 47 for more information.
[15:11]	-	Reserved, RES0

Bits	Name	Description
[10:0]	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[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

Accessibility

GICD_FCTLR2 is accessible only by Secure accesses.

5.2.8 GICD_UTILR, Utilization Register

This register controls the utilization engine in the LPI caches.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit

Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-8: GICD_UTILR bit assignments

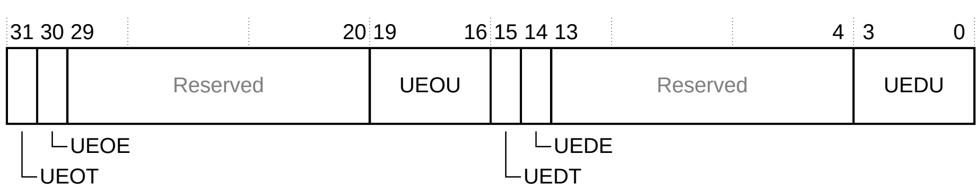


Table 5-10: GICD_UTILR bit descriptions

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	-
[3:0]	UEDU	Disabled utilization engine upper threshold. Automatically trigger the engine when the LPI cache bank is UEDU/16 full.	RW

5.2.9 GICD_RDOFFR, Redistributor Off Register

This register allows Secure software to remove up to 4 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 104 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 5-9: GICD_RDOFFR bit assignments

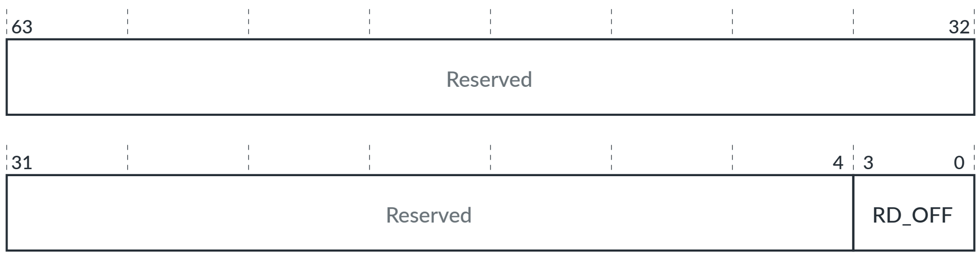


Table 5-11: GICD_RDOFFR bit descriptions

Bits	Name	Description
[63:4]	-	Reserved, RES0
[3:0]	RD_OFF	Controls whether a core is removed from the GIC: **Bit[m] = 0** Core m is not removed. **Bit[m] = 1** Removes core m. 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 234 for information about an MBIST limitation.

Accessibility

GICD_RDOFFR is accessible only by Secure accesses.

5.2.10 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-10: GICD_VCFGBASER bit assignments

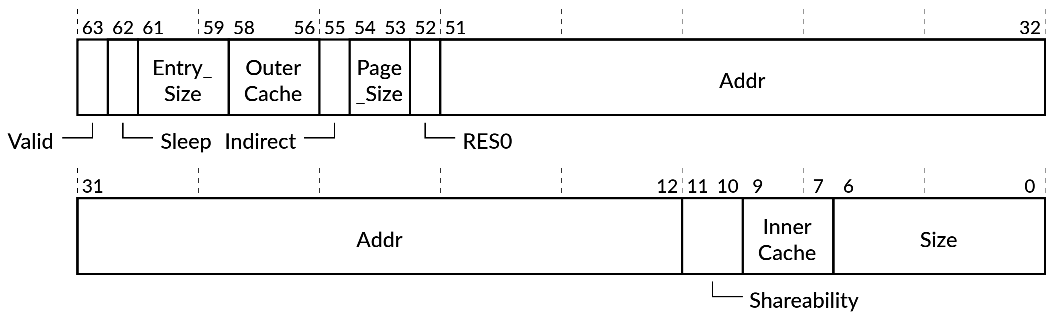


Table 5-12: GICD_VCFGBASER bit descriptions

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

5.2.11 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-11: GICD_VSLEEPR bit assignments

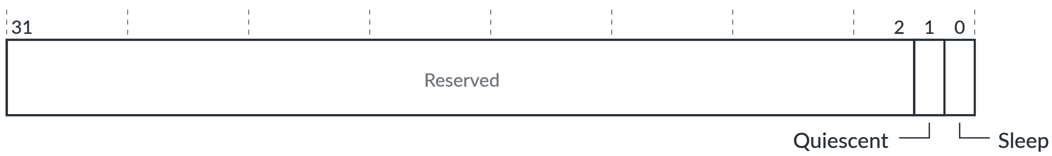


Table 5-13: GICD_VSLEEPR bit descriptions

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

5.2.12 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-700T 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 104 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-700T 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 5-12: GICD_ICLARn bit assignments

31 30 29 28	27 25 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Class							
n+15 | n+14	n+13 | n+12	n+11 | n+10	n+9 | n+8	n+7 | n+6	n+5 | n+4	n+3 | n+2	n+1 | n

Table 5-14: 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.

5.2.13 GICD_ICERRRn, Interrupt Clear Error Registers

Each register monitors 32 SPIs and the GIC-700T 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 104 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-700T 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 5-13: GICD_ICERRRn bit assignments

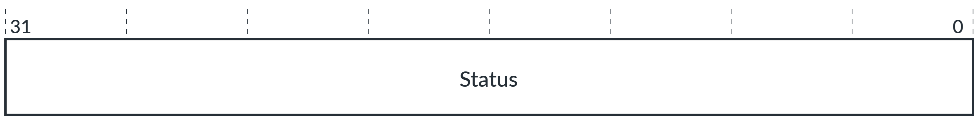


Table 5-15: GICD_ICERRRn bit descriptions

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].

5.2.14 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-700T 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 104 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-700T 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 5-14: GICD_ICGERRn bit assignments

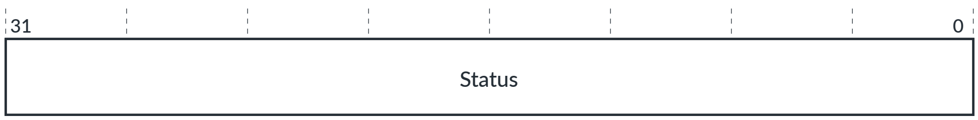


Table 5-16: GICD_ICGERRn bit descriptions

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.

5.2.15 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-700T has 30 registers, GICD_ISERRR1-GICD_ISERRR30. Software can use these registers to test the operation of its interrupt error clear function.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.2 Distributor registers (GICD/GICDA) summary on page 104 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-700T 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 5-15: GICD_ISERRRn bit assignments

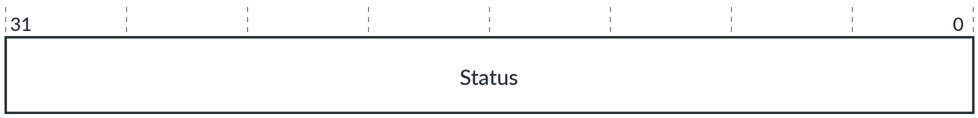


Table 5-17: GICD_ISERRRn bit descriptions

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.

5.2.16 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

See 4.4.1 RAM error simulation on page 44 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 5-18: 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.
[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 5-19: 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: **0** RAM is not present, or it is present but has no ECC. **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.

5.2.17 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-700T 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 104 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-700T 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 5-16: GICD_ICLARnE bit assignments

31 30	29 28	27 25	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Class															
n+15	n+14	n+13	n+12	n+11	n+10	n+9	n+8	n+7	n+6	n+5	n+4	n+3	n+2	n+1	n

Table 5-20: 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.

5.2.18 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-700T 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 104 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-700T 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 5-17: GICD_ICERRRnE bit assignments

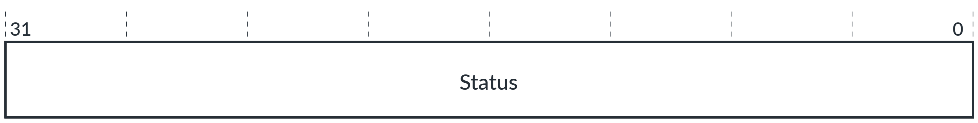


Table 5-21: GICD_ICERRRnE bit descriptions

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 5.2.14 GICD_ICGERRn, Interrupt Clear Group Error registers on page 123 or 5.2.19 GICD_ICGERRnE, Interrupt Clear Group Error registers Extended on page 128 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].

5.2.19 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-700T 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 104 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-700T 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 5-18: GICD_ICGERRnE bit assignments

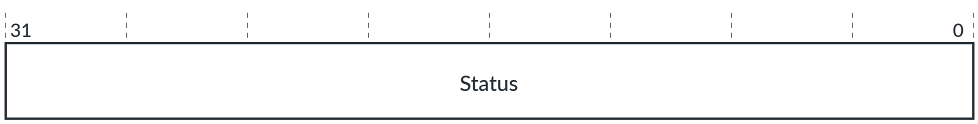


Table 5-22: GICD_ICGERRnE bit descriptions

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.

5.2.20 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-700T 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.

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 104 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-700T 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 5-19: GICD_ISERRRnE bit assignments

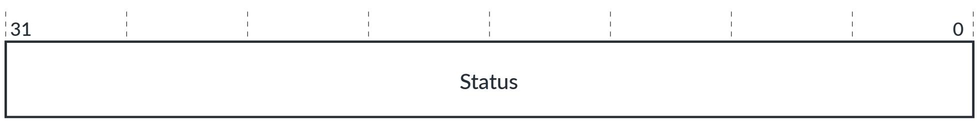


Table 5-23: GICD_ISERRRnE bit descriptions

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.

5.2.21 GICD_CFGID, Configuration ID Register

This register contains information that enables test software to determine if the GIC-700T 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-20: GICD_CFGID bit assignments

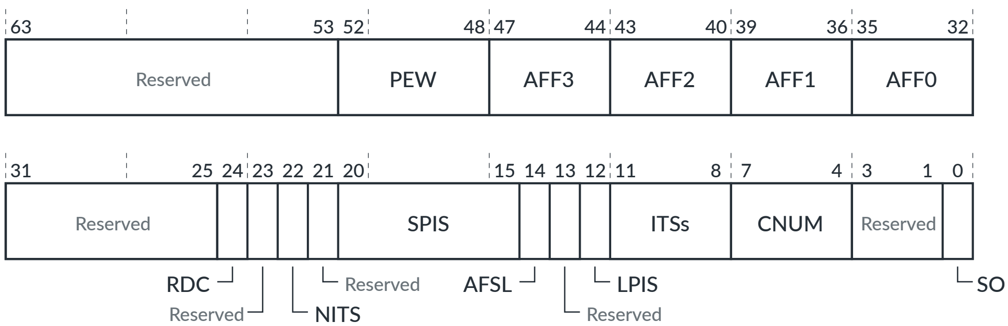


Table 5-24: GICD_CFGID bit assignments

Bits	Name	Description
[63:53]	-	Reserved, returns zero
[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 4.10.3 LPI operation on page 62 for more information.
[47:44]	AFF3	Returns the Affinity3 bits.
[43:40]	AFF2	Returns the Affinity2 bits.
[39:36]	AFF1	Returns the Affinity1 bits.
[35:32]	AFF0	Returns the Affinity0 bits.
[31:25]	-	Reserved, returns zero
[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 232 for more information.
[23]	-	Reserved, returns zero
[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).

Bits	Name	Description
[21]	-	Reserved, returns zero
[20:15]	SPIS	Number of SPI blocks supported.
[14]	-	Reserved
[13]	VLPIS	GICv4.1 supported
[12]	LPIS	LPI supported
[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).
[7:4]	CNUM	Chip number[3:0]
[3:1]	-	Reserved, returns zero
[0]	SO	Socket online status: **0** Single chip

Accessibility

The RDC bit is accessible only by Secure accesses.

5.2.22 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-21: GICD_PIDR4 bit assignments

Table 5-25: 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.

5.2.23 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-22: GICD_PIDR3 bit assignments

Table 5-26: 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`.

5.2.24 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-23: GICD_PIDR2 bit assignments

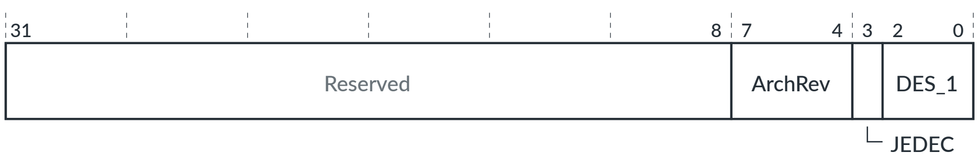


Table 5-27: GICD_PIDR2 bit descriptions

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].

5.2.25 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-24: GICD_PIDR1 bit assignments

Table 5-28: 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.

5.2.26 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 104 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-25: GICD_PIDR0 bit assignments

Table 5-29: 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-700T 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 5-30: 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	0x080nn43B 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 136 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-26: GICM_TYPER bit assignments

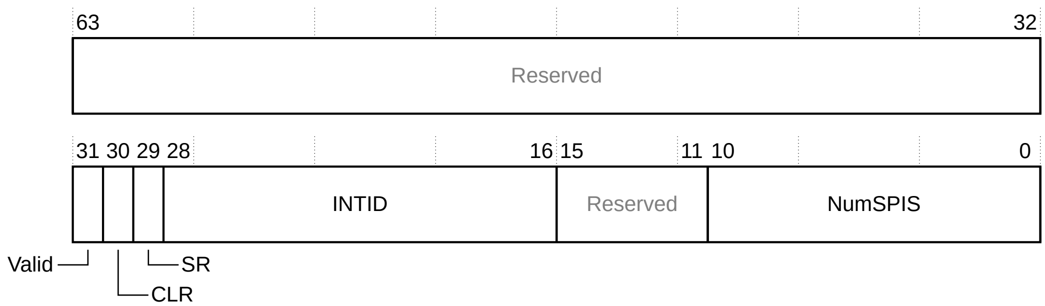


Table 5-31: GICM_TYPER bit descriptions

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 136 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-27: GICM_IIDR bit assignments

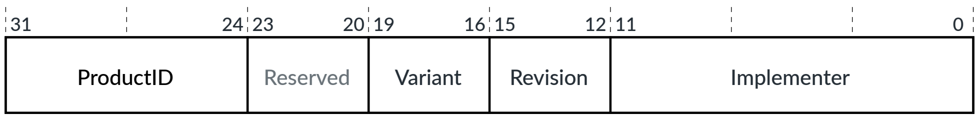


Table 5-32: GICM_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: `0x08` GIC-700T
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product `rxpy` identifier: `0x0` r0
[15:12]	Revision	Indicates the minor revision of the product `rxpy` identifier: `0x0` p0
[11:0]	Implementer	Identifies the implementer: `0x43B` Arm

5.4 Redistributor registers for control and physical LPIs summary

The functions for the GIC-700T 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 *Learn the architecture - Generic Interrupt Controller v3 and v4, LPIs*.

For descriptions of registers that are not specific to the GIC-700T, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Table 5-33: 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	`0x080nn43B` The `nn` value depends on the `rxpy` identifier.	32	Redistributor Implementation Identification Register	Yes
`0x0008`	GICR_TYPER	RO	Configuration dependent	64	Redistributor Type Register	Yes

Offset	Name	Type	Reset	Width	Description	Architecture defined?
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	Configuration dependent	32	Report maximum PARTID and PMG Register	Yes
0x001C	GICR_PARTIDR	RW	0x0	32	Set PARTID and PMG Register	Yes
0x0020	GICR_FCTLR	RW	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
0x002C-0x006C	-	-	-	-	Reserved	-
0x0070	GICR_PROPBASER	RW	Configuration dependent	64	Redistributor Properties Base Address Register. Only present when ITS and LPI 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 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	-
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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0xFFF8	GICR_CIDR2	RO	0x05	32	Component ID 2 Register	No
0xFFFC	GICR_CIDR3	RO	0xB1	32	Component ID 3 Register	No

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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-28: GICR_CTLR bit assignments

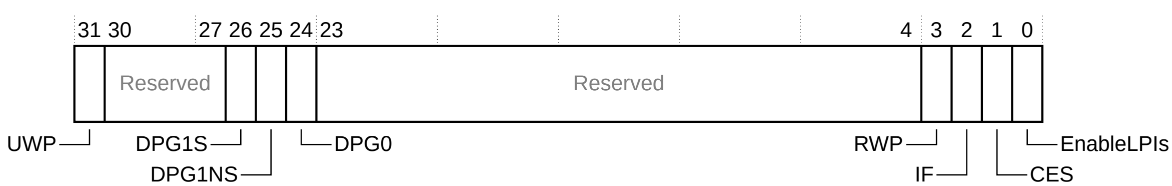


Table 5-34: GICR_CTLR bit descriptions

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	-
[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	-

Bits	Name	Description	Type
[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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-29: GICR_IIDR bit assignments

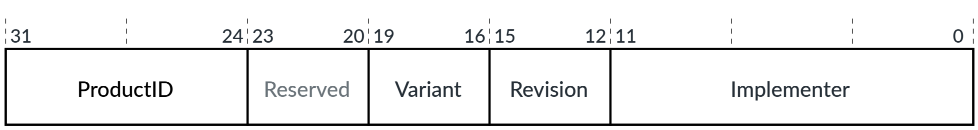


Table 5-35: GICR_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: `0x08` GIC-700T

Bits	Name	Description
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: **0x0** r0
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: **0x0** p0
[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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-30: GICR_TYPER bit assignments

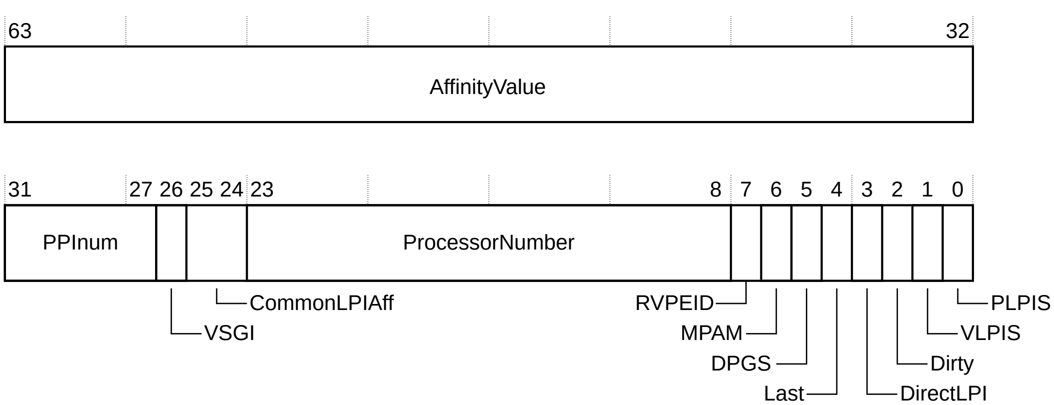


Table 5-36: GICR_TYPER bit descriptions

Bits	Name	Description
[63:32]	AffinityValue	Affinity level values for this Redistributor: **Bits[63:56], Aff3** The affinity level 3 value. **Bits[55:48], Aff2** The affinity level 2 value. **Bits[47:40], Aff1** The affinity level 1 value. **Bits[39:32], Aff0** The affinity level 0 value.
[31:27]	PPInum	Indicates the maximum PPI INTID that GIC-700T supports: `0b00000` Maximum PPI INTID is 31. `0b00001` Maximum PPI INTID is 1087.
[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.
[25:24]	CommonLPIAff	Returns: `0b00` Single chip configuration.
[23:8]	ProcessorNumber	Returns the core number and chip number that uniquely identifies this core in the system.
[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.
[6]	MPAM	Indicates whether GIC-700T 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.
[5]	DPGS	Returns 1, to indicate that the GIC-700T supports *Disable Processor Group Selections*. See GICR_CTLR.DPG1S, GICR_CTLR.DPG1NS, and GICR_CTLR.DPG0.
[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.
[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.
[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.

Bits	Name	Description
[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 *Learn the architecture - Generic Interrupt Controller v3 and v4, Virtualization*.
[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.

5.4.4 GICR_WAKER, Power Management Control Register

This register controls whether the GIC-700T 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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-31: GICR_WAKER bit assignments

Table 5-37: GICR_WAKER bit descriptions

Bits	Name	Description	Type
[31]	Quiescent	When set to 1, it indicates that the GIC-700T is idle and can be powered down if necessary. This bit indicates the GICs response to the Sleep bit, and it is only set to 1 when Sleep == 1.	RO
[30:3]	-	Reserved, RAZ	-

Bits	Name	Description	Type
[2]	ChildrenAsleep	When set to 1, it indicates that the bus between the CPU interface and this *GIC Cluster Interface* (GCI) is quiescent.	RO
[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 4.12.2 Processor core power management on page 67. 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 29.	RW
[0]	Sleep	Set this bit to 1, to flush the LPI cache: **0** Normal operation. **1** The GIC-700T 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 235. This bit is a separate control to the power controls that GICR_PWRR provides.	RW

Accessibility

GICR_WAKER is accessible only by Secure accesses.

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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-32: GICR_MPAMIDR bit assignments

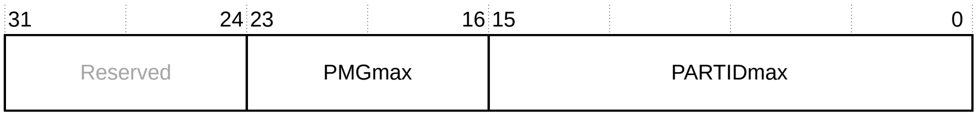


Table 5-38: GICR_MPAMIDR bit descriptions

Bits	Name	Description
[31:24]	-	Reserved
[23:16]	PMGmax	Performance monitoring group. Returns $2^{\texttt{pmg\_width}} - 1$, and indicates the maximum value that GICR_PARTIDR.PMG can be set to. `pmg_width` is a configuration parameter.
[15:0]	PARTIDmax	Returns $2^{\texttt{partid\_width}} - 1$, and indicates the maximum value that GICR_PARTIDR.PARTID can be set to. `partid_width` is a configuration parameter.

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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-33: GICR_PARTIDR bit assignments

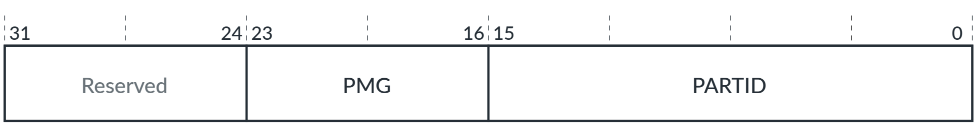


Table 5-39: GICR_PARTIDR bit descriptions

Bits	Name	Description
[31:24]	-	Reserved

Bits	Name	Description
[23:16]	PMG	The performance monitoring group value that the Redistributor uses when it accesses memory. The GIC allocates 8 bits for PMG, but GICR_MPAMIDR.PMGmax controls the usable width of this field, so some upper bits might be RES0.
[15:0]	PARTID	The Partition ID value that the Redistributor uses when it accesses memory. The GIC allocates 16 bits for PARTID, but GICR_MPAMIDR.PARTIDmax controls the usable width of this field, so some upper bits might be RES0.

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.

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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-34: GICR_FCTLR bit assignments

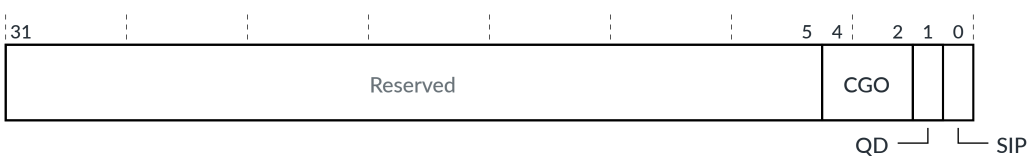


Table 5-40: GICR_FCTLR bit descriptions

Bits	Name	Description
[31:5]	-	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.

Bits	Name	Description
[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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-35: GICR_PWRR bit assignments

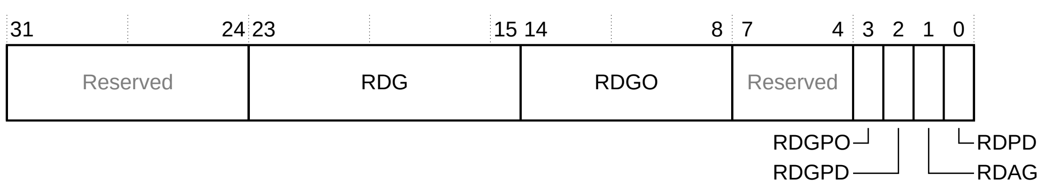


Table 5-41: GICR_PWRR bit descriptions

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

Bits	Name	Description	Type
[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. This bit changes state when the first or last PE on the GCI changes state.	RO
[2]	RDGPD	RDGroupPowerDown. This bit indicates the intentional power state of the GCI: **0** Intend to power up. **1** Intend to power down. This bit changes state when the first or last PE on the GCI changes state. 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
[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.

Related information

Redistributor power management on page 66

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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-36: GICR_CLASSR bit assignments

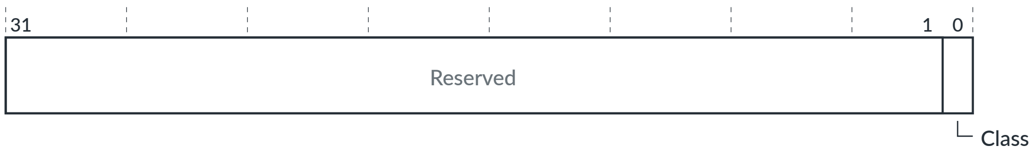


Table 5-42: GICR_CLASSR bit descriptions

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 54
GICD_ICLARn, Interrupt Class Registers on page 121

5.4.10 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 139 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.

Bit descriptions

Figure 5-37: GICR_MPIDR bit assignments

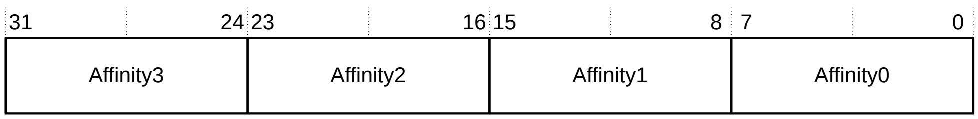


Table 5-43: GICR_MPIDR bit descriptions

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 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 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.

5.4.11 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 139 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-38: GICR_PIDR2 bit assignments

Table 5-44: 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-700T SGIs and PPIs are controlled through the Redistributor registers identified with the prefix GICR.

For descriptions of registers that are not specific to the GIC-700T, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Table 5-45: 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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0108-0x017C	-	-	-	-	Reserved	-
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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0D08-0x0DFC	-	-	-	-	Reserved	-
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-700T 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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-39: GICR_MISCSTATUSR bit assignments

Table 5-46: 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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-40: GICR_ICDERRR bit assignments

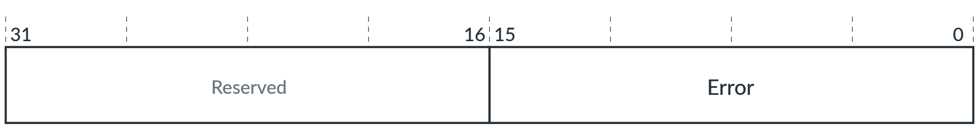


Table 5-47: GICR_ICDERRR bit descriptions

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).

Configurations

This register is available in all configurations. If SGI ECC is not enabled, then this register is RES0.

Attributes

Width 64-bit
Functional group See 5.5 Redistributor registers for SGIs and PPIs summary on page 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-48: 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.

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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-41: GICR_DPRIR bit assignments

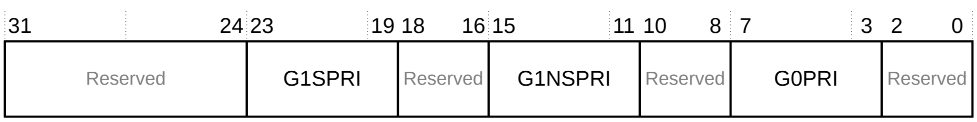


Table 5-49: GICR_DPRIR bit descriptions

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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-42: GICR_ICERRR0 bit assignments

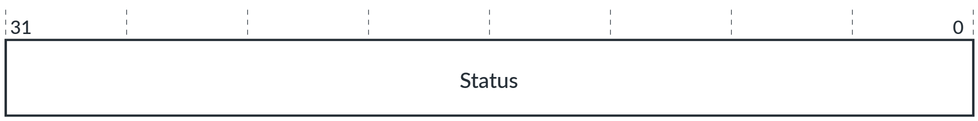


Table 5-50: GICR_ICERRR0 bit descriptions

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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-43: GICR_ICERRR1E bit assignments

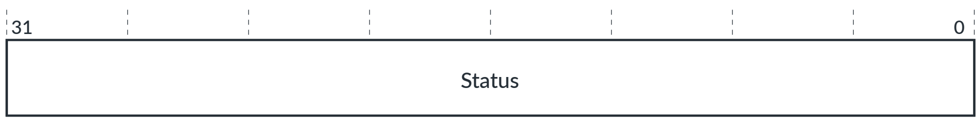


Table 5-51: GICR_ICERRR1E bit descriptions

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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-44: GICR_ISERRR0 bit assignments

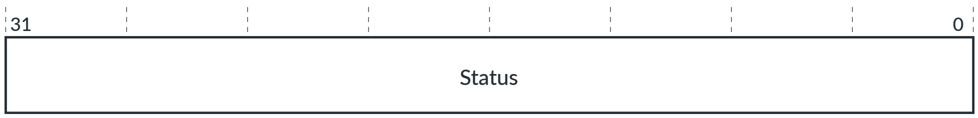


Table 5-52: GICR_ISERRR0 bit descriptions

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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-45: GICR_ISERRR1E bit assignments

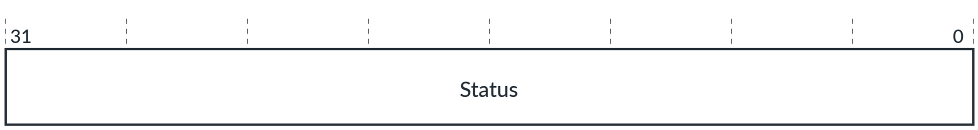


Table 5-53: GICR_ISERRR1E bit descriptions

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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-46: GICR_CFGID0 bit assignments

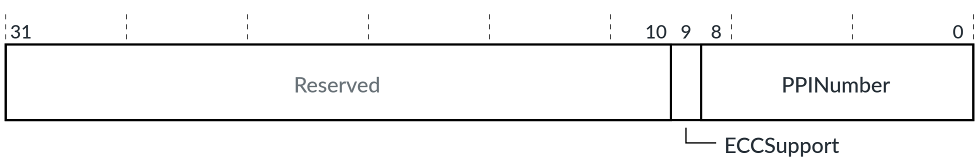


Table 5-54: GICR_CFGID0 bit descriptions

Bits	Name	Description
[31:10]	-	Reserved, RAZ
[9]	ECCSupport	1 = ECC is supported.
[8:0]	PPINumber	RedistributorID. Returns 0 because the GIC supports only 1 *GIC Cluster Interface* (GCI).

Related information

Miscellaneous signals on page 246

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 153 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-47: GICR_CFGID1 bit assignments

Table 5-55: GICR_CFGID1 bit descriptions

Bits	Name	Description
[31:28]	Version	Identifies the major and minor revisions of GIC-700T: **`0x0`** r0p0
[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-700T supports up to 4 cores, so the maximum value of this field is `0x03`.
[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 234 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 153 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 5-56: 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 5-57: 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: **0** RAM is not present, or it is present but has no ECC. **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.

5.6 vLPI register summary

The functions for the GIC-700T vLPIs are controlled through the Redistributor registers identified with the prefix GICR.

This page does not exist in GIC-700T configurations that do not support vLPIs.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for information about the vLPI registers.

Table 5-58: 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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
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.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit
Functional group See 5.6 vLPI register summary on page 166 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-48: GICR_VFCTLR bit assignments

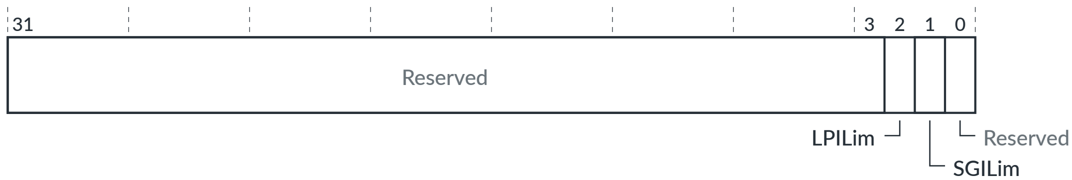


Table 5-59: GICR_VFCTLR bit descriptions

Bits	Name	Description
[31:3]	-	Reserved, **RES0**
[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]	-	Reserved, **RES0**

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 166 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-49: GICR_VCFGBASER bit assignments

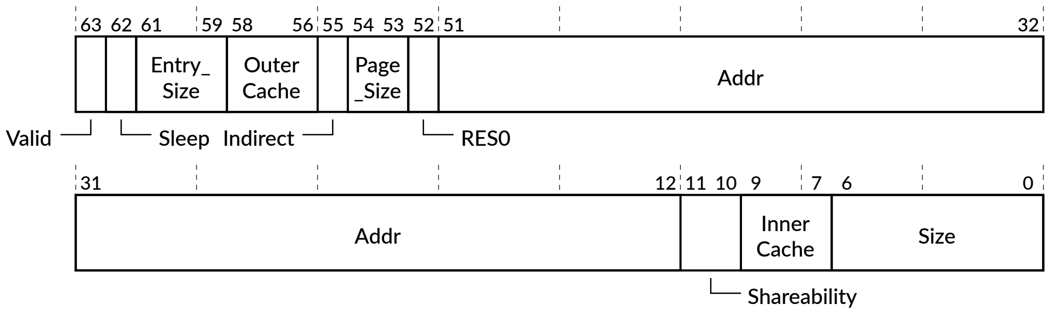


Table 5-60: GICR_VCFGBASER bit descriptions

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 is **RES0**.

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 166 for the address offset, type, and reset value of this register.

Usage constraints

This register is Reserved.

Bit descriptions

Table 5-61: GICR_VINVCHIPR bit descriptions

Bits	Name	Description
[31:0]	-	**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 166 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 5-62: 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

Bits	Name	Description	Type
[59:17]	-	**RES0**	-
[16:14]	Read_block	Controls which data to retrieve for an RD operation (Opcode == 0): **0** Doorbell data. See Table 5-63: GICR_VERRR bit assignments, for a Doorbell read request on page 171. **1** vPT data. See Table 5-64: GICR_VERRR bit assignments, for a vPT read request on page 172. **2, 5-7** vCONF data. See Table 5-65: GICR_VERRR bit assignments, for a vCONF read request on page 173. **3** vSGI[15:8] programming. See Table 5-66: GICR_VERRR bit assignments, for a vSGI read request on page 173. **4** vSGI[7:0] programming. See Table 5-66: GICR_VERRR bit assignments, for a vSGI read request on page 173.	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 5-63: 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.

Bits	Name	Description
[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 5-64: 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.
[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.
[57:43]	-	**RES0**
[42]	Mapped_ITS	Indicates if the vPE is mapped on the ITS: **0** The vPE is not mapped on the ITS. **1** The vPE is mapped on the ITS.
[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 5-65: 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: **0** The vPE is not mapped. **1** The vPE is mapped.
[58]	Errored	Indicates if the vPE is errored: **0** The vPE is not errored. **1** The vPE is errored.
[57:42]	-	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 5-66: 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

Bits	Name	Description
[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-700T *Interrupt Translation Service* (ITS) functions are controlled through registers that are identified with the prefix GITS.

This page does not exist in GIC-700T configurations that do not support LPIs.

For descriptions of registers that are not specific to the GIC-700T, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Table 5-67: 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	`0x080nn43B` The `nn` value depends on the `rxpy` identifier.	32	ITS Implementer Identification Register	Yes
`0x0008`	GITS_TYPER	RO	Configuration dependent	64	ITS Type Register	Yes
`0x0010`	GITS_MPAMIDR	RO	Configuration dependent	32	MPAM ID Register	Yes
`0x0014`	GITS_PARTIDR	RW	`0x0`	32	Part ID Register	Yes
`0x0018`	GITS_MPIDR	RO	`0x0`	32	ITS affinity	Yes
`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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0038-0x007C	-	-	-	-	Reserved	-
0x0080	GITS_CBASER	RW	0x0	64	ITS Command Queue Descriptor. See the *Learn the architecture - Generic Interrupt Controller v3 and v4, LPIs*.	Yes
0x0088	GITS_CWRITER	RW	0x0	64	ITS Write Register	Yes
0x0090	GITS_CREADR	RO	0x0	64	ITS Read 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
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 an ITS.

Attributes

Width 32-bit

Functional group See 5.7 ITS control register summary on page 174 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-50: GITS_IIDR bit assignments

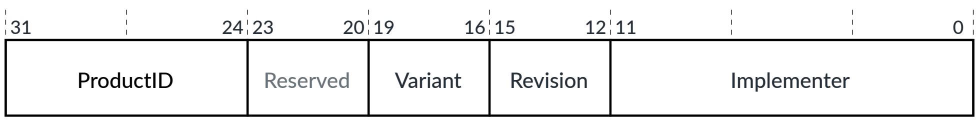


Table 5-68: GITS_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: `0x08` GIC-700T
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: `0x0` r0
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: `0x0` p0
[11:0]	Implementer	Identifies the implementer: `0x43B` Arm

5.7.2 GITS_TYPER, ITS Type Register

This register returns information about the features that the ITS supports.

Configurations

This register is available in all configurations that have an ITS.

Attributes

Width 64-bit

Functional group See 5.7 ITS control register summary on page 174 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-51: GITS_TYPER bit assignments

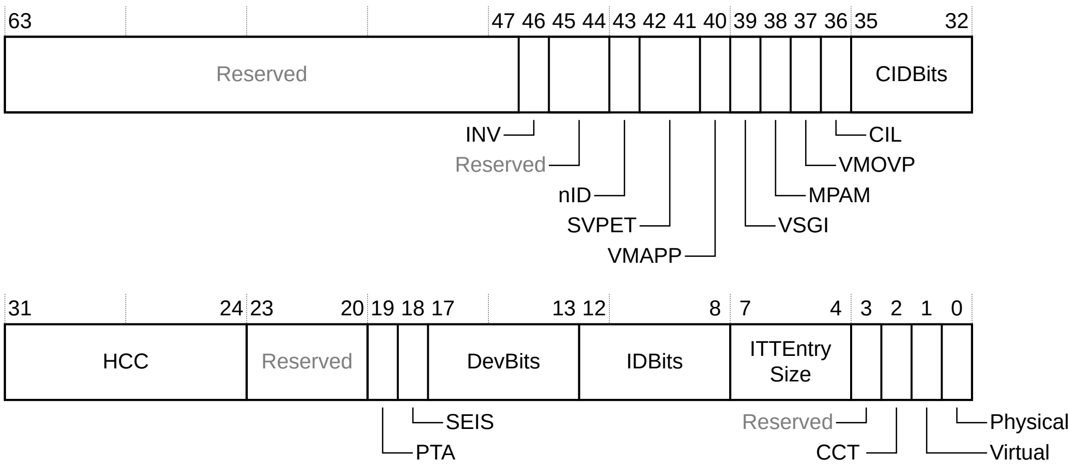


Table 5-69: GITS_TYPER bit descriptions

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-700T 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. 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 the ITS supports direct injection of SGIs: **0** The ITS does not support direct injection of SGIs. This value occurs when `gicv41_support` == 0. **1** The 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 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-700T does not support physical target addresses.
[18]	SEIS	System error interrupts: **0** The GIC-700T 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-700T 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 *Learn the architecture - Generic Interrupt Controller v3 and v4, Virtualization*.
[0]	Physical	Physical LPIs: **1** The GIC-700T 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 an ITS.

Attributes

Width 32-bit
Functional group See 5.7 ITS control register summary on page 174 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-52: GITS_MPAMIDR bit assignments

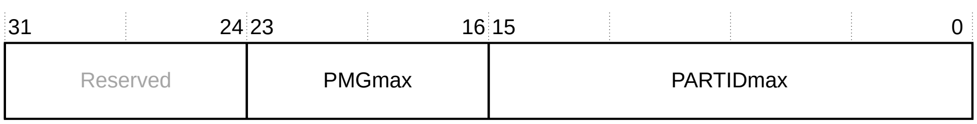


Table 5-70: GITS_MPAMIDR bit descriptions

Bits	Name	Description
[31:24]	-	Reserved
[23:16]	PMGmax	Performance monitoring group. Returns $2^{pmg\_width} - 1$, and indicates the maximum value that GITS_PARTIDR.PMG can be set to. `pmg_width` is a configuration parameter.
[15:0]	PARTIDmax	Returns $2^{partid\_width} - 1$, and indicates the maximum value that GITS_PARTIDR.PARTID can be set to. `partid_width` is a configuration parameter.

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 an ITS.

Attributes

Width 32-bit
Functional group See 5.7 ITS control register summary on page 174 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-53: GITS_PARTIDR bit assignments

Table 5-71: GITS_PARTIDR bit descriptions

Bits	Name	Description
[31:24]	-	Reserved
[23:16]	PMG	The performance monitoring group value that the ITS uses when it accesses memory. The GIC allocates 8 bits for PMG, but GITS_MPAMIDR.PMGmax controls the usable width of this field, so some upper bits might be RES0.
[15:0]	PARTID	The Partition ID value that the ITS uses when it accesses memory. The GIC allocates 16 bits for PARTID, but GITS_MPAMIDR.PARTIDmax controls the usable width of this field, so some upper bits might be RES0.

5.7.5 GITS_FCTLR, Function Control Register

This register controls many functions in the ITS such as cache invalidation, clock gating, and the scrubbing of all RAMs.

Configurations

This register is available in all configurations that have an ITS.

Attributes

Width 32-bit

Functional group See 5.7 ITS control register summary on page 174 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 5-54: GITS_FCTLR bit assignments

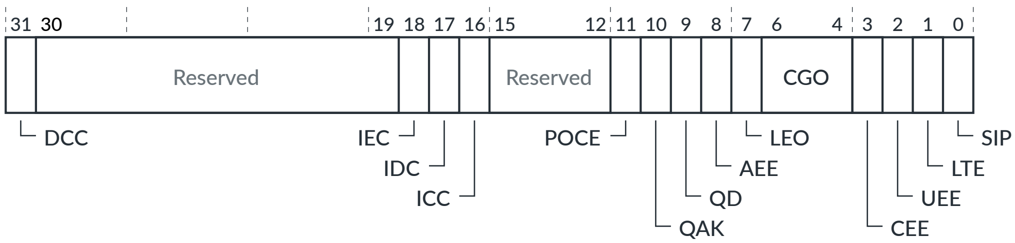


Table 5-72: GITS_FCTLR bit descriptions

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
[30:19]	-	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
[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

Bits	Name	Description	Type
[15:12]	-	Reserved, RAZ/WI	-
[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
[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 4.14.4.15 ITS command and translation error records 27 on page 87. 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

Bits	Name	Description	Type
[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 an ITS.

Attributes

Width 64-bit

Functional group See 5.7 ITS control register summary on page 174 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-55: GITS_OPR bit assignments

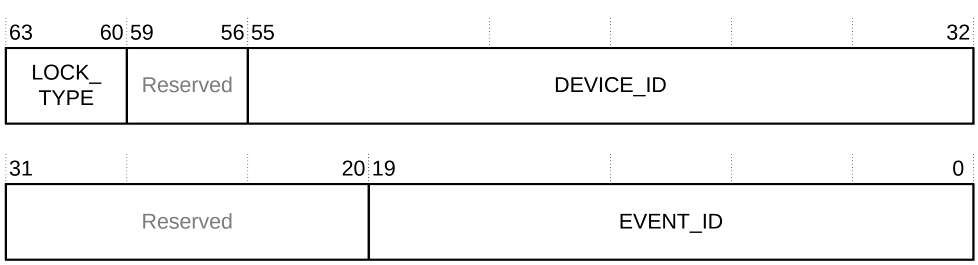


Table 5-73: GITS_OPR bit descriptions

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 an ITS.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 174 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-56: GITS_OPSR bit assignments

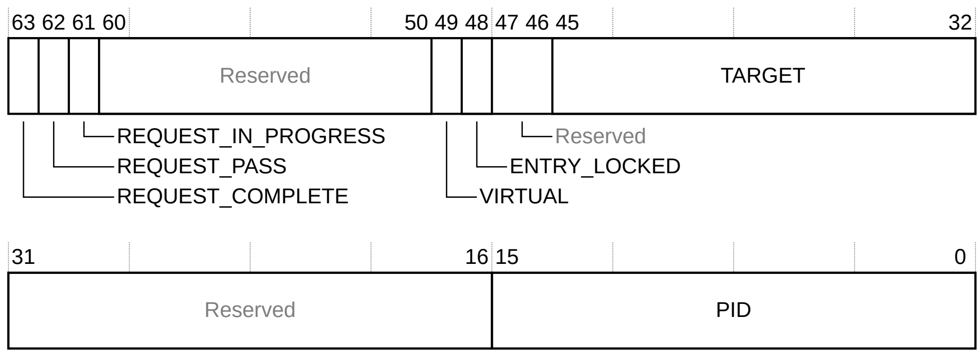


Table 5-74: GITS_OPSR bit descriptions

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**
[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 GIC-700T configurations when the ITS Device cache supports ECC.

Attributes

Width 64-bit

Functional group See 5.7 ITS control register summary on page 174 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 5-75: 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
[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 5-76: 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.

Bits	Name	Description
[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: **0** RAM is not present, or it is present but has no ECC. **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.

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 GIC-700T configurations when the ITS Event cache supports ECC.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 174 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 5-77: 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 5-78: 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: **0** RAM is not present, or it is present but has no ECC. **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.

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 GIC-700T configurations when the ITS Collection cache supports ECC.

Attributes

Width 64-bit

Functional group See 5.7 ITS control register summary on page 174 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 5-79: 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.

Bits	Name	Description
[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 5-80: 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: **0** RAM is not present, or it is present but has no ECC. **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_C_ERRINSR is accessible only by Secure accesses.

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 an ITS.

Attributes

Width 64-bit
Functional group See 5.7 ITS control register summary on page 174 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-57: GITS_CFGID bit assignments

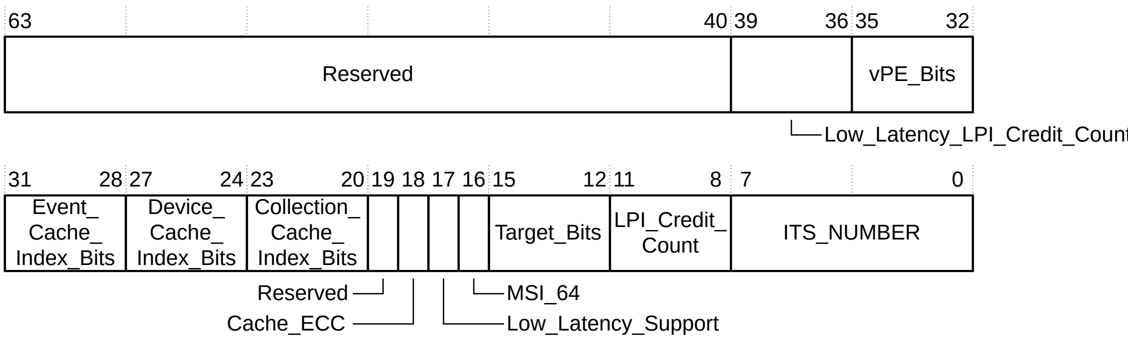


Table 5-81: GITS_CFGID bit descriptions

Bits	Name	Description
[63:40]	-	Reserved, RES0
[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. Returns 0 because the GIC supports only 1 ITS.

Related information

Miscellaneous signals on page 246

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 an ITS.

Attributes

Width 32-bit
Functional group See 5.7 ITS control register summary on page 174 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-58: GITS_PIDR2 bit assignments

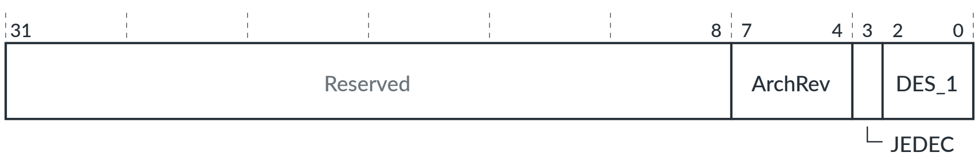


Table 5-82: GITS_PIDR2 bit descriptions

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-700T *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-700T configurations that do not support LPIs or that do not have an ITS.

Table 5-83: 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*.
`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-700T configurations that do not support vSGIs or that do not have an ITS.

Table 5-84: ITS vSGI register summary

Offset	Name	Type	Reset	Width	Description
`0x0000-0x001C`	-	-	-	-	Reserved
`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-700T 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 5-85: GICT register summary

Offset	Name	Type	Reset	Width	Description
`0x0000` + (n × 64)	GICT_ERR<n>FR	RO	Record dependent	64	Error Record Feature Register

Offset	Name	Type	Reset	Width	Description
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	0x080nn43B The nn value depends on the rxpy 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
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-700T implements.

Configurations

This register is available in all configurations.

Attributes

Width 32-bit

Functional group See 5.10 GICT register summary on page 193 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-59: GICT_ERR<n>FR bit assignments

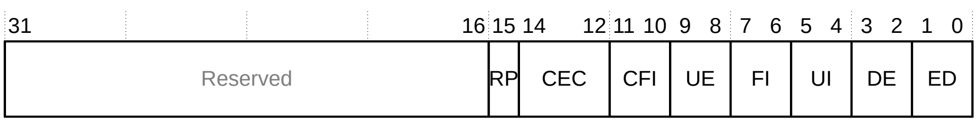


Table 5-86: GICT_ERR<n>FR bit descriptions

Bits	Name	Description
[31:16]	-	Reserved, RAZ
[15]	RP	Repeat corrected error count: **0** The GIC-700T does not implement a repeat corrected error counter.
[14:12]	CEC	Corrected error count: `0b000` The GIC-700T 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-700T does not provide a fault handling interrupt for corrected errors. `0b10` The GIC-700T provides a controllable fault handling interrupt for corrected errors.
[9:8]	UE	Uncorrected error. Depending on the configuration, returns either: `0b00` The GIC-700T does not provide an in-band uncorrected error reporting. `0b10` The GIC-700T 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-700T does not provide a fault handling interrupt. `0b10` The GIC-700T provides a controllable fault handling interrupt.

Bits	Name	Description
[5:4]	UI	Error recovery interrupt for uncorrected errors. Depending on the configuration, returns either: `0b00` The GIC-700T does not provide an error recovery interrupt for uncorrected errors. `0b10` The GIC-700T provides a controllable error recovery interrupt for uncorrected errors.
[3:2]	DE	Deferring of errors support: `0b00` The GIC-700T 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 193 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-60: GICT_ERR<n>CTLR bit assignments

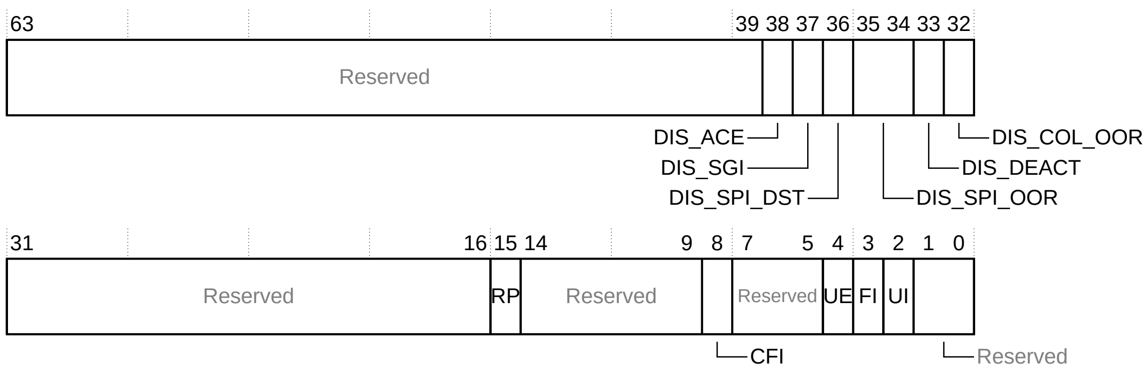


Table 5-87: GICT_ERR<n>CTLR bit descriptions

Bits	Name	Description
[63:39]	-	Reserved, RAZ
[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.
[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-700T does not assert a fault handling interrupt for corrected errors. **1** The GIC-700T 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 4.14.5 Bus errors on page 100.

Bits	Name	Description
[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 193 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-61: GICT_ERR<n>STATUS bit assignments

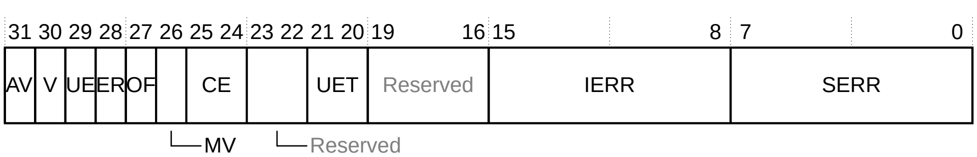


Table 5-88: GICT_ERR<n>STATUS bit descriptions

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 72: **`0b00`** No CE recorded. **`0b10`** At least one CE recorded.
[23:22]	-	Reserved, RAZ/WI
[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 5-91: GICT_ERR<n>MISC0.Data field encoding on page 202 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 5-91: GICT_ERR<n>MISC0.Data field encoding on page 202 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 193 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 5-62: GICT_ERR<n>ADDR bit assignments

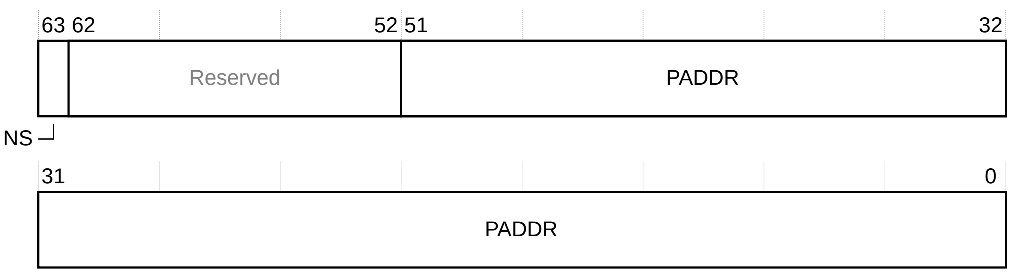


Table 5-89: GICT_ERR<n>ADDR bit descriptions

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 193 for the address offset, type, and reset value of this register.

Usage constraints

None

Bit descriptions

Figure 5-63: GICT_ERR<n>MISC0 bit assignments

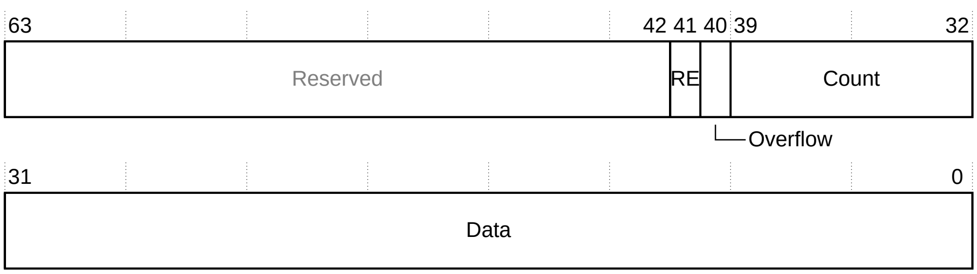


Table 5-90: GICT_ERR<n>MISC0 bit descriptions

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-700T 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.

Bits	Name	Description
[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 74 • Table 4-8: SPI RAM errors, records 1-2 on page 79 • Table 4-9: SGI RAM errors, records 3-4 on page 80 • Table 4-10: TGT_SPI RAM errors, records 5-6 on page 80 • Table 4-11: PPI RAM errors, records 7-8 on page 81 • Table 4-12: LPI RAM errors, records 9-10 on page 82 • Table 4-13: PTS RAM errors, records 11-12 on page 82 • Table 4-14: TGT_LPI RAM errors, records 13-14 on page 83 • Table 4-15: VICM RAM errors, records 15-16 on page 83 • Table 4-16: VSPA RAM errors, records 17-18 on page 84 • Table 4-17: VTGT_VSTR RAM errors, records 19-20 on page 85 • Table 4-18: VTGT_VRES RAM errors, records 21-22 on page 85 • Table 4-19: VTGT_SRCH RAM errors, records 23-24 on page 86 • Table 4-20: ITS RAM errors, records 25-26 on page 86 • 4.14.4.15 ITS command and translation error records 27 on page 87

The following table shows the Data field encoding for each error record and syndrome.

Table 5-91: 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)
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]
Software error (0)	`0x1`, SYN_PPI_PWRDWN Attempt to access a powered down Redistributor.	`0xF`	Redistributor, bits[24:16] Core, bits[8:0]
Software error (0)	`0x2`, SYN_PPI_PWRCHANGE Attempt to power down Redistributor rejected.	`0xF`	Redistributor, bits[24:16] Core, bits[8:0]
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]
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]
Software error (0)	`0x7`, SYN_WAKER_CHANGE Attempt to change GICR_WAKER abandoned due to handshake rules.	`0xF`	Core, bits[8:0]

<table>
<tr><th>Record</th><th>GICT_ERR<n>STATUS.IERR (syndrome)</th><th>GICT_ERR<n>STATUS .SERR</th><th>Value and description of GICT_ERR<n>MISC0.Data (other bits RES0)
Always packed from 0 (lowest = 0)</th></tr>
<tr><td>Software error (0)</td><td>0x8, SYN_SLEEP_FAIL
Attempt to put GIC to sleep failed because cores are not fully asleep.</td><td>0xF</td><td>Core, bits[8:0]</td></tr>
<tr><td>Software error (0)</td><td>0x9, SYN_PGE_ON_QUIESCE
Core put to sleep before its Group enables were cleared.</td><td>0xF</td><td>Core, bits[8:0]</td></tr>
<tr><td>Software error (0)</td><td>0x10, SYN_SGI_NO_TGT
SGI sent with no valid destinations.</td><td>0xE</td><td>Core, bits[8:0]</td></tr>
<tr><td>Software error (0)</td><td>0x11, SYN_SGI_CORRUPTED
SGI corrupted without effect.</td><td>0x6</td><td>Core, bits[8:0]</td></tr>
<tr><td>Software error (0)</td><td>0x12, SYN_GICR_CORRUPTED
Data was read from GICR register space that encountered an uncorrectable error.</td><td>0x6</td><td>GICT_ERR0ADDR is populated</td></tr>
<tr><td>Software error (0)</td><td>0x13, SYN_GICD_CORRUPTED
Data was read from GICD register space that encountered an uncorrectable error.</td><td>0x6</td><td>GICT_ERR0ADDR is populated</td></tr>
<tr><td>Software error (0)</td><td>0x14, SYN_ITS_OFF
Data was read from an ITS that is powered down.</td><td>0xF</td><td>GICT_ERR0ADDR is populated</td></tr>
<tr><td>Software error (0)</td><td>0x18, SYN_SPI_BLOCK.
Attempt to access an SPI block that is not implemented.</td><td>0xE</td><td>Block, bits[4:0]</td></tr>
<tr><td>Software error (0)</td><td>0x19, SYN_SPI_OOR
Attempt to access a non-implemented SPI using (SET|CLR)SPI.</td><td>0xE</td><td>ID, bits[9:0]</td></tr>
<tr><td>Software error (0)</td><td>0x1A, SYN_SPI_NO_DEST_TGT
An SPI has no legal target destinations.</td><td>0xF</td><td>ID, bits[9:0]</td></tr>
<tr><td>Software error (0)</td><td>0x1B, SYN_SPI_NO_DEST_1OFN
A 1 of N SPI cannot be delivered due to bad .DPG<0|1NS|1S> or GICR_CLASSR programming.</td><td>0xF</td><td>ID, bits[9:0]</td></tr>
<tr><td>Software error (0)</td><td>0x1C, SYN_COL_OOR
A collator message is received for a non-implemented SPI.</td><td>0xF</td><td>ID, bits[9:0]</td></tr>
<tr><td>Software error (0)</td><td>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.</td><td>0xE</td><td>None</td></tr>
<tr><td>Software error (0)</td><td>0x30, SYN_VSGI_UNMAPPED
Pending vSGI to a vPEID that is not mapped.</td><td>0xF</td><td>ID (multi-hot) [15:0]
vPEID[log$_2$(vpes)−1:0]</td></tr>
<tr><td>Software error (0)</td><td>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.</td><td>0x12</td><td>vPEID [log$_2$(vpes)−1:0]</td></tr>
<tr><td>Software error (0)</td><td>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.</td><td>0x12</td><td>vPEID [log$_2$(vpes)−1:0]</td></tr>
<tr><td>Software error (0)</td><td>0x39, SYN_VPE_CFG_PTR_FAIL
An attempt was made to access an indirect vPE Configuration table with an invalid level 2 pointer.</td><td>0xD</td><td>vPEID [log$_2$(vpes)−1:0]</td></tr>
</table>

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)
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]
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.	`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]
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]
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]
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]
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]
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]
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]
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]

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)
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]
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]
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]
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]
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]
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]
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]
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]
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]
Correctable SPI RAM errors (1)	**`0x00`** A real error **`0x01`** An injected error	`0x7`	See Table 4-8: SPI RAM errors, records 1-2 on page 79
Uncorrectable SPI RAM errors (2)	**`0x00`** A real error **`0x01`** An injected error	`0x7`	
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 80
Uncorrectable SGI RAM errors (4)	**`0x00`** A real error **`0x01`** An injected error	`0x7`	
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 80

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)
Uncorrectable TGT_SPI cache errors (6)	**0x00** **0x01**	A real error An injected error	0x7	
Correctable PPI RAM errors (7)	**0x00** **0x01**	A real error An injected error	0x7	See Table 4-11: PPI RAM errors, records 7-8 on page 81
Uncorrectable PPI RAM errors (8)	**0x00** **0x01**	A real error An injected error	0x7	
Correctable LPI RAM errors (9)	**0x00** **0x01**	A real error An injected error	0x7	See Table 4-12: LPI RAM errors, records 9-10 on page 82
Uncorrectable LPI RAM errors (10)	**0x00** **0x01**	A real error An injected error	0x7	
Correctable PTS RAM error (11)	**0x00** **0x01**	A real error An injected error	0x7	See Table 4-13: PTS RAM errors, records 11-12 on page 82
Uncorrectable PTS RAM error (12)	**0x00** **0x01**	A real error An injected error	0x7	
Correctable TGT_LPI RAM error (13)	**0x00** **0x01**	A real error An injected error	0x7	See Table 4-14: TGT_LPI RAM errors, records 13-14 on page 83
Uncorrectable TGT_LPI RAM error (14)	**0x00** **0x01**	A real error An injected error	0x7	
Correctable VICM RAM error (15)	**0x00** **0x01**	A real error An injected error	0x7	See Table 4-15: VICM RAM errors, records 15-16 on page 83
Uncorrectable VICM RAM error (16)	**0x00** **0x01**	A real error An injected error	0x7	
Correctable VSPA RAM error (17)	**0x00** **0x01**	A real error An injected error	0x7	See Table 4-16: VSPA RAM errors, records 17-18 on page 84
Uncorrectable VSPA RAM error (18)	**0x00** **0x01**	A real error An injected error	0x7	
Correctable VTGT_VSTR RAM error (19)	**0x00** **0x01**	A real error An injected error	0x7	See Table 4-17: VTGT_VSTR RAM errors, records 19-20 on page 85

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)
Uncorrectable VTGT_VSTR RAM error (20)	`0x00` A real error `0x01` An injected error	`0x7`	
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 85
Uncorrectable VTGT_VRES RAM error (22)	`0x00` A real error `0x01` An injected error	`0x7`	
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 86
Uncorrectable VTGT_SRCH RAM error (24)	`0x00` A real error `0x01` An injected error	`0x7`	
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 86
Uncorrectable error from ITS RAM (26)	`0x00` A real error `0x01` An injected error	`0x6`	
Command or translation error in ITS (27)	`0x00` Architectural `0x01` Non-architectural	`0x1`	ITS 24-bit syndrome. See 4.14.4.15 ITS command and translation error records 27 on page 87.

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 193 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 5-64: GICT_ERR<n>MISC1 bit assignments

Table 5-92: 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-700T.

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-700T 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 193 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-65: GICT_ERRGSR bit assignments

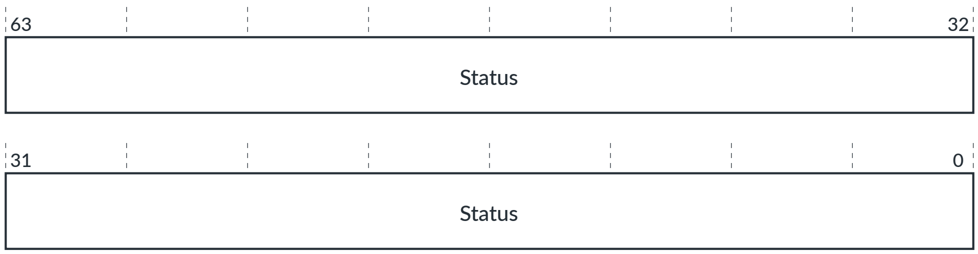


Table 5-93: GICT_ERRGSR bit descriptions

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 193 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-66: GICT_IIDR bit assignments

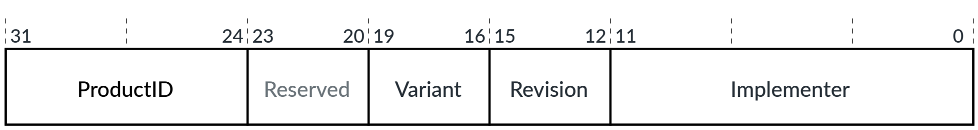


Table 5-94: GICT_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **`0x08`** GIC-700T
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product `rxpy` identifier: **`0x0`** r0
[15:12]	Revision	Indicates the minor revision of the product `rxpy` identifier: **`0x0`** p0
[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 193 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-67: GICT_ERRIRQCR<n> bit assignments

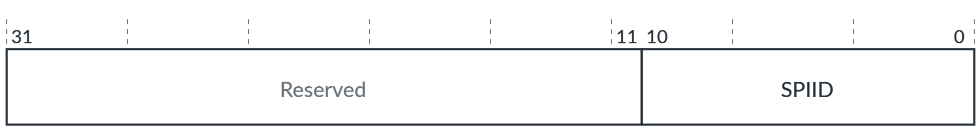


Table 5-95: GICT_ERRIRQCR<n> bit descriptions

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 or out of range, 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. 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.

5.10.10 GICT_DEVID, Device Configuration register

This register returns information about the configuration of the GIC-700T GICT such as whether an 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 193 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-68: GICT_DEVID bit assignments

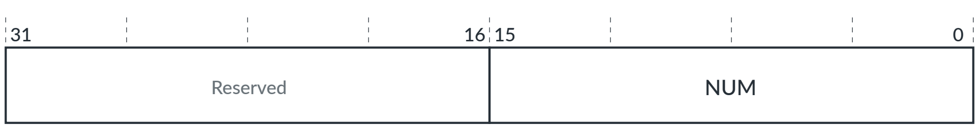


Table 5-96: GICT_DEVID bit descriptions

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** LPI available with one ITS.

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 193 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-69: GICT_PIDR2 bit assignments

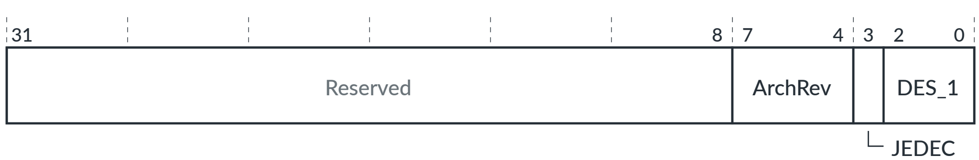


Table 5-97: GICT_PIDR2 bit descriptions

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.

Bits	Name	Description
[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-700T 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 5-98: 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
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	0x080nn43B The nn value depends on the rxpy 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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-70: GICP_EVCNTRn bit assignments

Table 5-99: 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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-71: GICP_EVTYPERn bit assignments

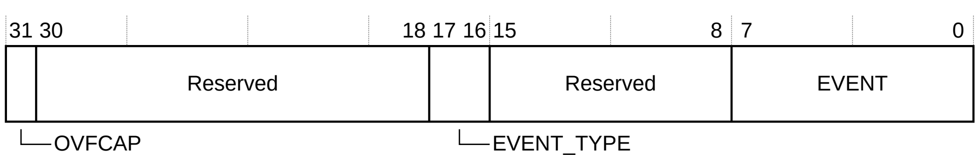


Table 5-100: GICP_EVTYPERn bit descriptions

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 5-101: GICP_EVTYPERn.EVENT field encoding on page 215. All events reset to an unknown value. Registers corresponding to unimplemented counters are RES0.

Table 5-101: GICP_EVTYPERn.EVENT field encoding

EventID	Event	Description	Mask	Filter
`0x0`	CLK	Clock cycle	Unmasked	None

EventID	Event	Description	Mask	Filter
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	TargetVP
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.	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.	TargetVP/ID range
0x7	-	Reserved	-	-
0x8	UP_MSG_PHY	Upstream message from core.	Masked	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.	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.	Target
0xB	UP_ACT_LPI	Upstream activate of LPI.	Unmasked	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.	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.	TargetVP/ID range
0xE	UP_ACT_DBL	Upstream activate of doorbell.	Unmasked	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.	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.	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.	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.	TargetVP/ID range
0x20	ITS_NLL_LPI	Incoming LPI	Unmasked	TargetVP/ID range/ITS
0x21	ITS_LL_LPI	Incoming low latency LPI.	Unmasked	TargetVP/ID range/ITS
0x22	ITS_LPI	Incoming LPI (or low latency).	Unmasked	TargetVP/ID range/ITS
0x23	ITS_LPI_CMD	Incoming LPI command	Unmasked	TargetVP/ID range/ITS

EventID	Event	Description	Mask	Filter
0x24	ITS_DID_MISS	Number of DeviceID cache misses.	Unmasked	TargetVP/ID range/ITS
0x25	ITS_VID_MISS	Number of EventID cache misses.	Unmasked	TargetVP/ID range/ITS
0x26	ITS_COL_MISS	Number of Collection cache misses.	Unmasked	TargetVP/ID range/ITS
0x27	ITS_LAT	Latency of the ITS transaction.	Unmasked	TargetVP/ID range/ITS
0x28	ITS_MPFA	Number of free slots during translation	Unmasked	TargetVP/ID range/ITS
0x30	LPI_OWN_STORED	LPI stored in own location. Prevents clock gating and Q-Channel clock gating.	Unmasked	-
0x31	LPI_OOL_STORED	LPI stored out of location. Prevents clock gating and Q-Channel clock gating.	Unmasked	-
0x32	LPI_HIT_EN	LPI property read cache hit enabled. Uses the filter from counter 0 only.	Unmasked	TargetVP/ID range
0x33	LPI_HIT_DIS	LPI property read cache hit disabled. Uses the filter from counter 0 only.	Unmasked	TargetVP/ID range
0x34	LPI_HIT	LPI property read cache hit. Uses the filter from counter 0 only.	Unmasked	TargetVP/ID range
0x35	LPI_MATCH	LPI coalesced. Uses the filter from counter 0 only.	Unmasked	TargetVP/ID range
0x36	LPI_FAS	Number of slots free on new LPI.	Unmasked	None
0x37	LPI_PROP_EN	Enabled LPI property fetch. Uses the filter from counter 0.	Unmasked	TargetVP/ID range
0x38	LPI_PROP_DIS	Disabled LPI property fetch. Uses the filter from counter 0.	Unmasked	TargetVP/ID range
0x39	LPI_PROP	LPI property fetch. Uses the filter from counter 0.	Unmasked	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.	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.	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.	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.	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.	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.	ID range

EventID	Event	Description	Mask	Filter
0x60	PT_IN_EN	Enabled interrupt written to Pending table.	Unmasked	TargetVP/ID range
0x61	PT_IN_DIS	Disabled interrupt written to Pending table.	Unmasked	TargetVP/ID range
0x62	PT_PRI	Priority of interrupt written to Pending table.	Unmasked	TargetVP/ID range
0x63	PT_IN	Interrupt written to Pending table.	Unmasked	TargetVP/ID range
0x64	PT_MATCH	Interrupt already set in Pending table.	Unmasked	TargetVP/ID range
0x65	PT_OUT_EN	Enabled interrupt taken out of Pending table (also covered PT_MATCH when enabled).	Unmasked	TargetVP/ID range
0x66	PT_OUT_DIS	Disabled interrupt taken out of Pending table (also covered PT_MATCH when disabled).	Unmasked	TargetVP/ID range
0x67	PT_OUT	Interrupt taken out of Pending table (also covered PT_MATCH).	Unmasked	TargetVP/ID range
0x72	VSGI_IN_RAM	vSGI stored in RAM.	Unmasked	TargetVP
0x73	VLPI_BUFF_FILL	Number of buffers used on vLPI arriving.	Unmasked	-
0x78	RES_START	Residency change start.	Unmasked	TargetVP
0x79	RES_COMP	Residency change end.	Unmasked	TargetVP
0x80	ACC	Counter(n − 1) − counter(n − 2) every cycle. Prevents clock gating and Q-Channel clock gating.	Unmasked	None
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	TargetVP(PHY)/ ID range
0x89	UP_REL_VIRT	Virtual release	Unmasked	TargetVP(PHY)
0x8A	UP_ACT_VLPI	Activate of vLPI.	Unmasked	TargetVP(PHY)/ ID range
0x8B	UP_ACT_VSGI	Activate of vSGI.	Unmasked	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	Target(PHY)

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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-72: GICP_SVRn bit assignments

Table 5-102: 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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-73: GICP_FRn bit assignments

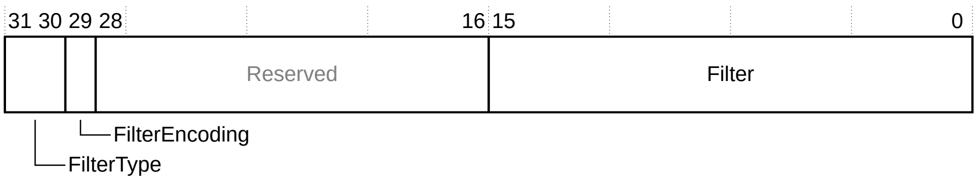


Table 5-103: GICP_FRn bit descriptions

Bits	Name	Description
[31:30]	FilterType	Filter type: **0b00** Filter on core or vPE or both. **0b01** Filter on INTID. **0b10** Filter on ITS. **0b11** Reserved, no effect.
[29]	FilterEncoding	**0** Filter on range. **1** Filter on an exact match.
[28:16]	-	Reserved
[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 5-101: GICP_EVTYPERn.EVENT field encoding on page 215, 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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-74: GICP_CNTENSET0 bit assignments

Table 5-104: 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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-75: GICP_CNTENCLR0 bit assignments

Table 5-105: 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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-76: GICP_INTENSET0 bit assignments

Table 5-106: GICP_INTENSET0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ
[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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-77: GICP_INTENCLR0 bit assignments

Table 5-107: GICP_INTENCLR0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ
[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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-78: GICP_OVSCLR0 bit assignments

Table 5-108: GICP_OVSCLR0 bit descriptions

Bits	Name	Description
[31:5]	-	Reserved, RAZ
[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 215.

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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-79: GICP_OVSSET0 bit assignments

Table 5-109: 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-700T 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-80: GICP_CAPR bit assignments

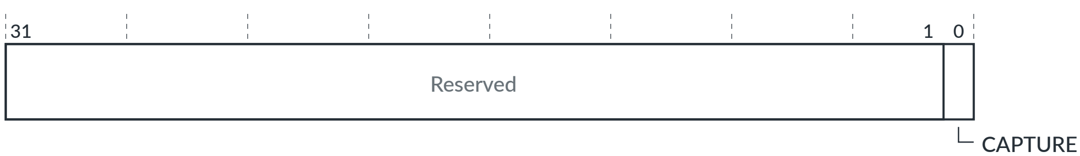


Table 5-110: GICP_CAPR bit descriptions

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 70 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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-81: GICP_CFGR bit assignments

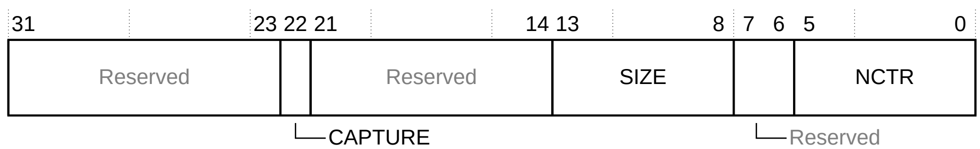


Table 5-111: GICP_CFGR bit descriptions

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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-82: GICP_CR bit assignments

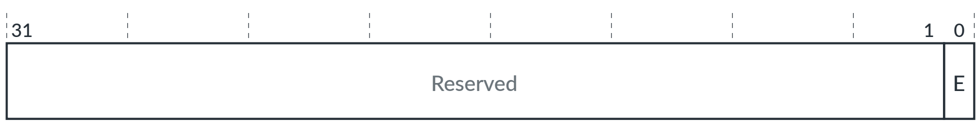


Table 5-112: GICP_CR bit descriptions

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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-83: GICP_IIDR bit assignments

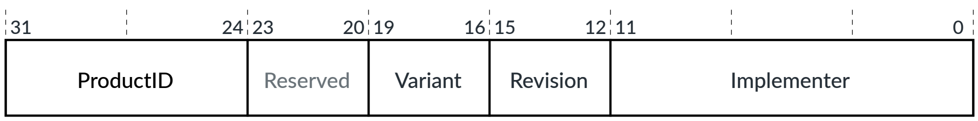


Table 5-113: GICP_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: `0x08` GIC-700T
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision, or variant, of the product r*x*p*y* identifier: `0x0` r0
[15:12]	Revision	Indicates the minor revision of the product r*x*p*y* identifier: `0x0` p0
[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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-84: GICP_IRQCR bit assignments

Table 5-114: 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 or out of range, 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. Otherwise it behaves as a normal message-based SPI.

Accessibility

If GICD_SAC.GICPNS == 0, then GICP_IRQCR is accessible only by Secure accesses.

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 213 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-85: GICP_PIDR2 bit assignments

Table 5-115: 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.

Appendix A Getting started with GIC-700T

There are some basic tasks that you must complete before you can start to use GIC-700T.

Each Redistributor must be powered on using its GICR_PWRR register to enable the Redistributors to be accessed, see 4.12.1 Redistributor power management on page 66 for more information.

When the GIC-700T is powered up, it must be programmed as the *Learn the architecture - Generic Interrupt Controller v3 and v4, Overview* 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. 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 and sets a bit to 1 to remove that core from the configuration. For example, to remove:
   - The 1st core, set GICD_RDOFFR[0] to 1.
   - The 2nd core, set GICD_RDOFFR[1] to 1.
   - The 4th core, set GICD_RDOFFR[3] 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 233.
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_RDOFFR 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_RDOFFR 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_RDOFFR, 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.

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. If this behavior is an issue for the system, then we recommend that the GCI RAM is implemented as flops without ECC.

A.2 Other power management

The GIC-700T can be powered up and powered down using non-architectural protocols.

When powering up GIC-700T, then software must program registers in the following sequence:

1. If using programmable core removal, program GICD_RDOFFRn and then GICR_MPIDR.
2. Any other registers.

When powering down GIC-700T, software must preserve the state of the GIC-700T, 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-700T without deadlock.

When the GICR_WAKER.Quiescent bit is set, it is safe to power down the GIC-700T 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-700T if any interrupts arrive that must not be ignored.

When the GIC-700T 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-700T 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.EnableLPIs 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 - Generic Interrupt Controller v3 and v4, Overview*.

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<n>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 or out of range, 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.

Appendix B Signal descriptions for GIC-700T

This appendix describes the external input and output signals of the GIC-700T.

B.1 Common control signals

The following table shows the GIC-700T common control signal set.

Signal definitions

Table B-1: Common control signals

Signal	Direction	Description
[<domain>]clk	Input	Clock input
[<domain>]reset_n	Input	Active-LOW reset
dbg_[<domain>]reset_n	Input	Active-LOW reset for the PMU and error records. This signal is only present for the domain that contains the Distributor.

Test signals		
Signal	**Direction**	**Description**
dftrstdisable	Input	Reset disable. Disables the external reset input for test mode. When this signal is HIGH, it forces the internal active-LOW reset HIGH, bypassing the reset synchronizer.
dftse	Input	Scan enable. Disables clock gates for test mode.
dftcgen	Input	Clock gate enable. When this signal is HIGH, it forces all the clock gates on so that all internal clocks always run.
dftramhold	Input	RAM hold. When this signal is HIGH, it forces all the RAM chip selects LOW, preventing accesses to the RAMs.

MBIST controller signals		
Signal	**Direction**	**Description**
[<domain>_]mbistack	Output	MBIST mode ready. GIC-700T acknowledges that it is ready for MBIST testing.
[<domain>_]mbistreq	Input	MBIST mode request. Request to GIC-700T to enable MBIST testing. This signal must be tied LOW during functional operation.
[<domain>_]nmbistreset	Input	Resets MBIST logic. Resets functional logic to enable MBIST operation by an active-LOW signal. This signal must be tied HIGH during functional operation.
[<domain>_]mbistaddr[n:0]	Input	Logical address. The width is based on the RAM with the largest number of words, which depends on the configuration. You must drive the most significant bits to zero when accessing RAMs with fewer address bits.
[<domain>_]mbistindata[n:0]	Input	Data in. Write data. Width that is based on the RAM with the largest number of data bits, which depends on the configuration.

MBIST controller signals		
Signal	**Direction**	**Description**
[<domain>_]mbistoutdata[n:0]	Output	Data out. Read data. Width that is based on the RAM with the largest number of data bits, which depends on the configuration.
[<domain>_]mbistwriteen	Input	Write control (mbistwriteen) and read control (mbistreaden). No access occurs if both enables are LOW. It is illegal to activate both enables simultaneously.
[<domain>_]mbistreaden	Input	
[<domain>_]mbistarray[n:0]	Input	Array selector. This signal controls which RAM array is accessed. The signal width depends on the configuration. For the single RAM configuration, this port is unused. This signal is not present on a block containing only one RAM.
[<domain>_]mbistcfg	Input	MBIST ALLMODE enable. When enabled, allows simultaneous access to all RAM arrays for maximum array power consumption. This signal is not present on a block containing only one RAM.

B.2 Power control signals

The following table shows the GIC-700T power control signals.

Signal definitions

Table B-2: Power control signals

Signal	Direction	Description
cpu_active[<cpus>−1:0]	Input	Indicates if the core is active and not in a low-power state such as retention. This signal is used for lowering the priority of selection for 1 of N SPIs. There is 1 bit for each core on the ICC bus. See 4.12.2 Processor core power management on page 67.
wake_request[<cpus>−1:0]	Output	Wake Request signal to power controller indicating that an interrupt is targeting this core and it must be woken. When asserted, the wake_request signal is sticky unless the Distributor is put into the gated state.
cpu_wake_request[<gci_cpus>−1:0]	Output	Wake request signal to a core, indicating that an interrupt is targeting the core and it must be woken. When HIGH, the cpu_wake_request signal is sticky unless the GICR_PWRR.RDGPD powerdown bit is set.
spi_ram_retained	Input	When HIGH, it informs the GICD that the SPI programming in the SPI RAMs has been retained during powerdown. The GIC samples the value as it exits reset. See 4.12.3 SPI RAM retention on page 68 for more information.

SPI Collator Q-Channel device interfaces for power control		
Signal	**Direction**	**Description**
	Input	Q-Channel device interface to flush out the path between the SPI Collator and the Distributor to aid in power down. When asserted, messages are not sent to the Distributor until low-power state is exited. If the GIC contains two or more SPI Collators, then `<n>` is a numeric identifier for an SPI Collator. For example, if the GIC has two SPI Collators, then `<n>` is 0 or 1. **Note:** It is only safe to stop the SPI Collator clock if all interrupts are level sensitive, or if edge-triggered interrupts are pulse extended into the SPI Collator.
	Output	
	Output	
	Output	

Distributor Q-Channel device interface for clock control		
Signal	**Direction**	**Description**
qreqn	Input	Q-Channel device interface for clock gating of the Distributor. The qreqn signal is synchronized into the GIC-700T. This bus must be treated asynchronously.
qacceptn	Output	
qdeny	Output	
qactive	Output	

GCI Q-Channel device interface for clock control		
Signal	**Direction**	**Description**
qreqn	Input	Q-Channel device interface for clock gating of a GCI. The qreqn signal is synchronized into the GIC-700T. This bus must be treated asynchronously.
qacceptn	Output	
qdeny	Output	
qactive	Output	

Q-Channel device interfaces for clock control		
Signal	**Direction**	**Description**
[<domain_>]clkqreqn	Input	Q-Channel device interface for clock gating of everything in the domain. The [<domain_>]clkqreqn signal is synchronized into the GIC-700T. This bus must be treated asynchronously.
[<domain_>]clkqacceptn	Output	
[<domain_>]clkqdeny	Output	
[<domain_>]clkqactive	Output	

Q-Channel ADB-400 device interfaces for power control		
Signal	**Direction**	**Description**
[<domain_>]pwrqreqn	Input	Q-Channel device interface for the CoreLink™ ADB-400 AMBA® Domain Bridge power interface within the domain.
[<domain_>]pwrqacceptn	Output	
[<domain_>]pwrqdeny	Output	
[<domain_>]pwrqactive	Output	

B.3 Interrupt signals

The GIC-700T has interrupt signals for PPIs and SPIs.

Signal definitions

Table B-3: Interrupt signals

Signal	Direction	Description
ppi<n>[<cpus>−1:0] If there are: • 16 PPIs for each core, `n` is 16-31. • 32 PPIs for each core, `n` is 16-31 and 1056-1071. • 48 PPIs for each core, `n` is 16-31 and 1056-1087.	Input	PPI input wires for interrupt <n>. One bit for each core. The PPIs for each core are independent and are typically used for peripherals that are not shared between cores. For example, timers on the core typically use PPIs. By default, PPIs are active-LOW. The GIC provides build-time options so that a PPI can be active-HIGH. The GIC also provides build-time options so that a PPI can be synchronized to the clk signal. By default, PPIs are level-sensitive interrupts. However, software can change an interrupt to be edge triggered by programming the GICR_ICFGR1, GICR_ICFGR2E, and GICR_ICFGR3E registers.
ppi<n>_r	Output	PPI output after synchronization and edge detection. You can use these signals to create pulse extenders for edge-triggered interrupts that cross clock domains.
spi[spi_wire−1:0] The `spi_wire` configuration parameter controls the number of SPIs.	Input	This signal is the number of SPI wires that the GIC supports. **Note:** This is not the same as the number of SPIs supported because they could be message-based only. By default, SPIs are active-HIGH. The GIC provides build-time options so that an SPI can be active-LOW. The GIC also provides build-time options so that an SPI can be synchronized to the clk signal.
spi_r[spi_wire−1:0] The `spi_wire` configuration parameter controls the number of SPIs.	Output	SPI output after synchronization and edge detection. Can be used for cross-domain pulse detection. The `SPI_R_INV` build-time option can remove any inversion that `SPI_INV[n]` applies to individual SPIs on that SPI Collator. See 3.5.2 SPI Collator wires on page 35.

B.4 CPU interface signals

The CPU interface signals of a cluster connect to a Redistributor using two GIC Stream interfaces. A Redistributor is also known as a *GIC Cluster Interface* (GCI).

In the following tables, <cpuif_stream_width>, and <cpus> are configuration options that are set using the `cpuif_stream_width`, and `cpus` parameters. See the *Arm® CoreLink™ GIC-700T Generic Interrupt Controller Configuration and Integration Manual* for more information.

Signal definitions

Table B-4: CPU interface signals

GIC Stream-compliant bus for communication from a cluster to a Redistributor		
Signal	**Direction**	**Description**
icctready	Output	This GIC Stream-compliant bus is fully credited and can be sent over any free-flowing interconnect. For more information, see *Table A-2 CPU interface to upstream Redistributor interface* in the *GIC Stream Protocol interface Appendix* of the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*. If the cluster issues IDs on the ICCTID signal with values other than `<cpus-1:0>`, then the behavior is unpredictable.
icctvalid	Input	
icctdata[<cpuif_stream_width>−1:0]	Input	
icctid[<cpus>−1:0]	Input	
icctlast	Input	
icctwakeup	Input	Registered wake signal to indicate that a message is arriving or is about to arrive on the icc bus

GIC Stream-compliant bus for communication from a Redistributor to a cluster		
Signal	**Direction**	**Description**
iritready	Input	This GIC Stream-compliant bus is fully credited and can be sent over any free-flowing interconnect. For more information, see *Table A-1 Redistributor to downstream CPU interface* in the *GIC Stream Protocol interface Appendix* of the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.
iritvalid	Output	
iritdata[<cpuif_stream_width>−1:0]	Output	
iritdest [<cpus>−1:0]	Output	
iritlast	Output	
iritwakeup	Output	Registered wake signal to indicate that a message is arriving or is about to arrive on the IRI bus of the cluster

B.5 ACE5-Lite interface signals

The following table shows the GIC-700T ACE5-Lite signals.

Table B-5: ACE5-Lite subordinate interface signals

Subordinate wakeup signal		
Signal	**Direction**	**Description**
awakeup_s	Input	Interface wake up signal

Subordinate write address channel signals		
Signal	**Direction**	**Description**
awvalid_s	Input	Write address valid
awready_s	Output	Write address ready
awid_s[n:0]	Input	Write address ID, where `n = axis_wid_width-1`
awaddr_s[n:0]	Input	Write address, where `n = axis_addr_width-1`
awlen_s[7:0]	Input	Write burst length
awsize_s[2:0]	Input	Write burst size
awburst_s[1:0]	Input	Write burst type
awprot_s[2:0]	Input	Write protection type

Subordinate write address channel signals		
Signal	**Direction**	**Description**
awcache_s[3:0]	Input	Write cache type
awuser_s[n:0]	Input	Optional User signal. Where, `n` = `axis_awuser_width`-1. Indicates the *DeviceID* of writes to GITS_TRANSLATER if MSI_64 is not configured.
awsnoop_s[n:0]	Input	Indicates the transaction snoop type. `n` == 3 when `axi_invalidate_hint_support`==0 and `n` == 4 when `axi_invalidate_hint_support`==1.
awdomain_s[1:0]	Input	Indicates the shareability domain
awtrace_s	Input	Trace signal
awloop_s[n:0]	Input	Loopback signal, where `n` = `axis_awloop_width`-1
awidunq_s	Input	Write address unique ID indicator
awnse_s	Input	Write root/realm. Only present when RME_Support is True.
awmpam_s[n:0]	Input	Write MPAM signal. `n` depends on the values of the `axi_rme_support`, `pmg_width`, and `partid_width` parameters.
awtagop_s[1:0]	Input	Write request tag operation. Only present when MTE_Support is True.

Subordinate write data channel signals		
Signal	**Direction**	**Description**
wvalid_s	Input	Write data valid
wready_s	Output	Write data ready
wdata_s[n:0]	Input	Write data, where `n` = `axis_data_width`-1
wstrb_s[n:0]	Input	Write data byte lane strobes
wtag_s[n:0]	Input	Write data tag, where `n` depends on `axis_data_width`. Only present when MTE_Support is True.
wtagupdate_s[n:0]	Input	Write data tag update indicator, where `n` depends on `axis_data_width`. Only present when MTE_Support is True.
wlast_s	Input	Write data last transfer indicator
wtrace_s	Input	Trace signal
wpoison_s[n:0]	Input	Poison signal, where `n` varies depending on the ACE5-Lite interface

Subordinate write response channel signals		
Signal	**Direction**	**Description**
bvalid_s	Output	Write response valid
bready_s	Input	Write response ready
bidunq_s	Output	Write response unique ID indicator
bid_s[n:0]	Output	Write response ID, where `n` = `axis_wid_width`-1
bresp_s[1:0]	Output	Write response
buser_its_<num>_s[n:0]	Output	Write response User signal, where `n` = `axis_buser_width`-1
btrace_s	Output	Trace signal
bloop_s[n:0]	Output	Loopback signal, where `n` = `axis_awloop_width`-1

Subordinate read address channel signals		
Signal	**Direction**	**Description**
arvalid_s	Input	Read address valid
arready_s	Output	Read address ready

Subordinate read address channel signals		
Signal	**Direction**	**Description**
arid_s[n:0]	Input	Read address ID, where `n` = `axis_rid_width-1`
araddr_s[n:0]	Input	Read address, where `n` = `axis_addr_width-1`
arlen_s[7:0]	Input	Read burst length
arsize_s[2:0]	Input	Read burst size
arburst_s[1:0]	Input	Read burst type
arprot_s[2:0]	Input	Read protection type
arcache_s[3:0]	Input	Read cache type
arsnoop_s[3:0]	Input	Indicates the transaction snoop type
ardomain_s[1:0]	Input	Indicates the shareability domain
arlock_s	Input	Read lock type
archunken_s	Input	Chunk enable signal. If asserted, read data for this transaction can be returned out of order, in 128-bit chunks.
artagop_s[1:0]	Input	Read request tag operation. Only present when MTE_Support is True.
aruser_s	Input	This signal indicates some user-defined sideband content that transfers with the read address. The GIC-700T ignores aruser data that it receives on the GICD (Distributor) subordinate port.
aridunq_s	Input	Read address unique ID indicator
arnse_s	Input	Read root/realm. Only present when RME_Support is True.
artrace_s	Input	Trace signal
arloop_s[n:0]	Input	Loopback signal, where `n` = `axis_arloop_width-1`
armpam_s[n:0]	Input	Read MPAM signal. `n` depends on the values of the `axi_rme_support`, `pmg_width`, and `partid_width` parameters.

Subordinate read data channel signals		
Signal	**Direction**	**Description**
rvalid_s	Output	Read data valid
rready_s	Input	Read data ready
rid_s[n:0]	Output	Read data ID, where `n` = `axis_rid_width-1`
rdata_s[n:0]	Output	Read data, where `n` = `axis_data_width-1`
rresp_s[1:0]	Output	Read data response
rtag_s[n:0]	Output	Read data tag, where `n` depends on `axis_data_width`. Only present when MTE_Support is True.
rlast_s	Output	Read data last transfer indicator
rtrace_s	Output	Trace signal
rloop_s[n:0]	Output	Loopback signal, where `n` = `axis_arloop_width-1`
rpoison_s[n:0]	Output	Poison signal, where `n` varies depending on the ACE5-Lite interface
ridunq_s	Output	Read data unique ID indicator
rchunkv_s	Output	If asserted, rchunknum_s and rchunkstrb_s are valid for this transfer
rchunknum_s[n:0]	Output	Indicates the number of chunks being transferred. Chunks are numbered incrementally from zero, according to the data width and base address of the transaction. `n` = `CHUNKNUM_WIDTH-1`.
rchunkstrb_s[n:0]	Output	Indicates which part of read data is valid for this transfer. Each bit corresponds to 128 bits of data: **rchunkstrb[0]** Corresponds to rdata[127:0] **rchunkstrb[1]** Corresponds to rdata[255:128]

The following table shows the GIC-700T ACE5-Lite manager signals. This interface is present only when the GIC is configured to supports LPIs.

Table B-6: ACE5-Lite manager interface signals

Manager wakeup signal. Only present if LPI support is configured.		
Signal	**Direction**	**Description**
awakeup_m	Input	Interface wake up signal

Manager write address channel signals. Only present if LPI support is configured.		
Signal	**Direction**	**Description**
awvalid_m	Output	Write address valid
awready_m	Input	Write address ready
awid_m[n:0]	Output	Write address ID. `n` is calculated from various elements, during the configuration process.
awaddr_m[n:0]	Output	Write address, where `n = axim_addr_width-1`
awlen_m[7:0]	Output	Write burst length
awsize_m[2:0]	Output	Write burst size
awburst_m[1:0]	Output	Write burst type
awprot_m[2:0]	Output	Write protection type
awcache_m[3:0]	Output	Write cache type
awsnoop_m[3:0]	Output	Indicates the transaction snoop type
awdomain_m[1:0]	Output	Indicates the shareability domain
awidunq_m	Output	Write address unique ID indicator
awnse_m	Output	Write root/realm. Only present when RME_Support is True.
awmpam_m[n:0]	Output	Write MPAM signal. `n` depends on the values of the `axi_rme_support`, `pmg_width`, and `partid_width` parameters.

Manager write data channel signals. Only present if LPI support is configured.		
Signal	**Direction**	**Description**
wvalid_m	Output	Write data valid
wready_m	Input	Write data ready
wdata_m[n:0]	Output	Write data, where `n = axim_data_width-1`
wstrb_m[n:0]	Output	Write data byte lane strobes
wlast_m	Output	Write data last transfer indicator
wpoison_m[n:0]	Output	Poison signal, where `n` varies depending on the ACE5-Lite interface

Manager write response channel signals. Only present if LPI support is configured.		
Signal	**Direction**	**Description**
bvalid_m	Input	Write response valid
bready_m	Output	Write response ready
bidunq_m	Input	Write response unique ID indicator
bid_m[n:0]	Input	Write response ID. `n` is calculated during the configuration process, from various elements.
bresp_m[1:0]	Input	Write response
bloop_m[n:0]	Input	Loopback signal, where `n = axis_awloop_width-1`

Manager read address channel signals. Only present if LPI support is configured.		
Signal	**Direction**	**Description**
arvalid_m	Output	Read address valid
arready_m	Input	Read address ready
arid_m[n:0]	Output	Read address ID. `n` is calculated from various elements, during the configuration process.
araddr_m[n:0]	Output	Read address, where `n = axim_addr_width-1`
arlen_m[7:0]	Output	Read burst length
arsize_m[2:0]	Output	Read burst size
arburst_m[1:0]	Output	Read burst type
arprot_m[2:0]	Output	Read protection type
arcache_m[3:0]	Output	Read cache type
arsnoop_m[3:0]	Output	Indicates the transaction snoop type
ardomain_m[1:0]	Output	Indicates the shareability domain
arnse_m	Output	Read root/realm. Only present when RME_Support is True.
archunken_m	Output	Chunk enable signal. If asserted, read data for this transaction can be returned out of order, in 128-bit chunks.
aridunq_m	Output	Read address unique ID indicator
armpam_m[n:0]	Output	Read MPAM signal. `n` depends on the values of the `axi_rme_support`, `pmg_width`, and `partid_width` parameters.

Manager read data channel signals. Only present if LPI support is configured.		
Signal	**Direction**	**Description**
rvalid_m	Input	Read data valid
rready_m	Output	Read data ready
rid_m[n:0]	Input	Read data ID. `n` is calculated from various elements, during the configuration process.
rdata_m[n:0]	Input	Read data, where `n = axim_data_width-1`
rresp_m[1:0]	Input	Read data response
rlast_m	Input	Read data last transfer indicator
rpoison_m[n:0]	Input	Poison signal, where `n` varies depending on the ACE5-Lite interface
rchunkv_m	Input	If asserted, rchunknum_m and rchunkstrb_m are valid for this transfer.
rchunknum_m[n:0]	Input	Indicates the number of chunks being transferred. Chunks are numbered incrementally from zero, according to the data width and base address of the transaction. `n = CHUNKNUM_WIDTH-1`.
rchunkstrb_m[n:0]	Input	Indicates which part of read data is valid for this transfer. `n = axim_data_width/128-1`. Each bit corresponds to 128 bits of data. For example: **rchunkstrb[0]** Corresponds to rdata[127:0] **rchunkstrb[1]** Corresponds to rdata[255:128]
ridunq_m	Input	Read data unique ID indicator

B.6 Miscellaneous signals

The following table shows the GIC-700T miscellaneous signals.

Signal definitions

Table B-7: Miscellaneous signals

Signal	Direction	Description
ppi_id[15:0]	Input	An ID number that identifies the *GIC Cluster Interface* (GCI) in the system. Software can read the GICR_CFGID0 register to access the value of this signal.
its_id[7:0]	Input	An ID number that identifies the ITS block in the system. Software can read the GITS_CFGID register to access the value of this signal.
fault_int	Output	Fault handling interrupt. The fault handling interrupt is defined in the *RAS System Architecture* chapter of the *Arm® Architecture Reference Manual for A-profile architecture*. The GIC-700T can deliver this interrupt internally but the output is provided for any other device such as a system control processor that does not receive normal interrupts from the GIC. See 4.14.3 Error recovery and fault handling interrupts on page 71.
err_int	Output	Error handling interrupt. The error handling interrupt is defined in the *RAS System Architecture* chapter of the *Arm® Architecture Reference Manual for A-profile architecture*. The GIC-700T can deliver this interrupt internally but the output is provided for any other device such as a system control processor that does not receive normal interrupts from the GIC. See 4.14.3 Error recovery and fault handling interrupts on page 71.
pmu_int	Output	PMU counter overflow interrupt. This signal is a level-sensitive interrupt. The GIC-700T can deliver this interrupt internally but the output is provided as an external output to trigger an external agent to service the GIC, for example, to read out the PMU counter snapshot registers. See Overflow interrupt on page 70.
sample_req	Input	Request from a *Cross Trigger Interface* (CTI) to sample the PMU counters. Equivalent to writing to the GICP_CAPR register. See Snapshot on page 70 for more information.
sample_ack	Output	This signal goes HIGH when the GIC acknowledges the PMU sample request from the CTI
gict_allow_ns	Input	From reset, this tie-off signal controls whether Non-secure software can access the GICT Error Record registers
gicp_allow_ns	Input	From reset, this tie-off signal controls whether Non-secure software can access the GICP PMU registers
gicd_ctlr_ds	Input	From reset, this tie-off signal controls whether the GIC supports both Security states: • LOW = Security is enabled. The GIC supports both Security states. • HIGH = Security is disabled. The GIC supports a single Security state. Software can read the GICD_CTLR.DS bit to access the value of this signal.
gicd_page_offset	Input	From reset, this tie-off signal controls the page address bits[x:16] of the GICD page. This signal is present only in monolithic configurations. See Page offset on page 102.
gicd_pe_off[max_pe_on_chip – 1:0]	Input	From reset, this tie-off signal controls which cores are removed from the GIC configuration. The `max_pe_on_chip` parameter sets the number of cores that the chip supports. Set a bit to 1, to remove the corresponding core. The behavior is unpredictable when all bits are set to 1. This signal is present only when `prog_mpidr = strap`
affinity0[(max_pe_on_chip × max_affinity_width0) – 1:0]	Input	From reset, this tie-off signal sets the affinity 0 value for each core. This signal is present only when `prog_mpidr = strap` and `max_affinity_width0` is nonzero.
affinity1[(max_pe_on_chip × max_affinity_width1) – 1:0]	Input	From reset, this tie-off signal sets the affinity 1 value for each core. This signal is present only when `prog_mpidr = strap` and `max_affinity_width1` is nonzero.
affinity2[(max_pe_on_chip × max_affinity_width2) – 1:0]	Input	From reset, this tie-off signal sets the affinity 2 value for each core. This signal is present only when `prog_mpidr = strap` and `max_affinity_width2` is nonzero.

Signal	Direction	Description
its_transr_page_offset	Input	From reset, this tie-off signal controls the page address of the GITS_TRANSLATER register. This signal is present only in monolithic configurations. See 4.9.2 MSI-64 on page 59 and Page offset on page 102.
msi_translator_page	Input	The target page address of the GITS_TRANSLATER register. The MSI-64 Encapsulator does not support an msi_transalator_page signal value of 0. See 3.4 MSI-64 Encapsulator on page 32.
msi64_translator_page	Input	The target address of the 64-bit GITS_TRANSLATER register. This page must be at a different location to the msi_translator_page signal and at a location that is known only to the hypervisor. The hypervisor must be able to protect the page from accesses from devices and processors that can spoof incorrect DeviceIDs. See 3.4 MSI-64 Encapsulator on page 32 and 4.9.2 MSI-64 on page 59.
awdeviceid	Input	An ACE5-Lite AW sideband signal that reports the DeviceID for writes to GITS_TRANSLATER. The value is ignored for non-MSI writes, which are writes that do not target the 64KB region that starts from the address that the msi64_translator_page signal selects. The MSI-64 uses this signal only when `AWDEVICEID_FROM_AWUSER` == 0. See 3.4 MSI-64 Encapsulator on page 32 and 3.4.1 MSI-64 ACE5-Lite interfaces on page 33.
spi_base[10:0]	Input	This signal sets the base address of an SPI Collator. This signal is present only when the GIC is configured to use signals rather than parameters to set the base address of each SPI Collator. See 3.5.6 SPI Collator configuration on page 37.

B.7 RAM I/O signals

The GIC can be configured to provide sideband I/O signals to each RAM. You can use the I/O to control elements within your RAM models.

The RAM I/O signals are present when the GIC is configured to support the RAM I/O signals. See 3.1.5 Distributor configuration on page 27.

Signal definitions

Table B-8: RAM I/O signals

Signal	Direction	Description
ci_ram_in[CI_RAM_IN_WIDTH−1:0]	Input	These I/O signals have no inherent functionality inside the GIC.
ci_ram_out[CI_RAM_OUT_WIDTH−1:0]	Output	
dcache_ram_in[DCACHE_RAM_IN_WIDTH−1:0]	Input	
dcache_ram_out[DCACHE_RAM_OUT_WIDTH−1:0]	Output	
vcache_ram_in[VCACHE_RAM_IN_WIDTH−1:0]	Input	
vcache_ram_out[VCACHE_RAM_OUT_WIDTH−1:0]	Output	
ccache_ram_in[CCACHE_RAM_IN_WIDTH−1:0]	Input	
ccache_ram_out[CCACHE_RAM_OUT_WIDTH−1:0]	Output	
vicm_ram_in[VICM_RAM_IN_WIDTH−1:0]	Input	
vicm_ram_out[VICM_RAM_OUT_WIDTH−1:0]	Output	
vtgt_store_ram_in[VTGT_VSTR_RAM_IN_WIDTH−1:0]	Input	
vtgt_store_ram_out[VTGT_VSTR_RAM_OUT_WIDTH −1:0]	Output	
vtgt_search_ram_in[VTGT_SRCH_RAM_IN_WIDTH−1:0]	Input	

Signal	Direction	Description
vtgt_search_ram_out[VTGT_SRCH_RAM_OUT_WIDTH −1:0]	Output	
vtgt_residency_ram_in[VTGT_VRES_RAM_IN_WIDTH −1:0]	Input	
vtgt_residency_ram_out[VTGT_VRES_RAM_OUT_WIDTH −1:0]	Output	
vspa_ram_in[VSPA_RAM_IN_WIDTH−1:0]	Input	
vspa_ram_out[VSPA_RAM_OUT_WIDTH−1:0]	Output	
sgi_ram_in[SGI_RAM_IN_WIDTH−1:0]	Input	
sgi_ram_out[SGI_RAM_OUT_WIDTH−1:0]	Output	
tgt_spi_ram_in[TGT_SPI_RAM_IN_WIDTH−1:0]	Input	
tgt_spi_ram_out[TGT_SPI_RAM_OUT_WIDTH−1:0]	Output	
spi<n>_ram_in[SPI_RAM_IN_WIDTH−1:0]	Input	These I/O signals have no inherent functionality inside the GIC. <n> ranges from 0 upwards, and the number of signals depends on the channel count.
spi<n>_ram_out[SPI_RAM_OUT_WIDTH−1:0]	Output	
lpi<0-3>_ram_in[LPI_RAM_IN_WIDTH−1:0]	Input	These I/O signals have no inherent functionality inside the GIC.
lpi<0-3>_ram_out[LPI_RAM_OUT_WIDTH−1:0]	Output	
pts_ram_in[PTS_RAM_IN_WIDTH−1:0]	Input	
pts_ram_out[PTS_RAM_OUT_WIDTH−1:0]	Output	
tgt_lpi_ram_in[TGT_LPI_RAM_IN_WIDTH−1:0]	Input	
tgt_lpi_ram_out[TGT_LPI_RAM_OUT_WIDTH−1:0]	Output	

B.8 Interdomain signals

Interdomain signals are routed between domains.

Signal definitions

Table B-9: Interdomain signals

Signal	Direction	Description
wakeup_sm_*	Input and output, depends on signal name	These signals connect between halves of a CoreLink™ ADB-400. See the *Arm® CoreLink™ ADB-400 AMBA® Domain Bridge User Guide*. If you instantiate domain levels, you must ensure that matching input and output pairs of interdomain signals connect together directly, and are not separated by synchronizers.
wakeup_ms_*		
*_async_*		

Appendix C Implementation-defined features of GIC-700T

The GIC-700T 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-700T. 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-700T uses. The table also gives references to sections within this manual that provide information about implementation-defined behavior that is specific to the GIC-700T.

Table C-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 4.8.3 SPI routing and 1 of N selection on page 54
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 4.14 Reliability, Accessibility, and Serviceability on page 71.
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.
		Interrupt prioritization	GIC-700T supports 32 priority levels, 16 for LPIs that are always Non-secure.

GICv4.1 architecture feature	Architectural specification reference		Description
	Chapter	Section	
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 4.10.4 LPI caching on page 62 and 4.9 ITS on page 57.
LPI Configuration tables	Locality-specific peripheral interrupts and the ITS	LPI Configuration tables	The GIC-700T has one GICR_PROPBASER register for all cores on a chip and therefore points to a single table.
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 D Revisions

This appendix describes the technical changes between released issues of this document.

The first table is for the first release. Then, each table compares the new issue of the manual with the last released issue of the manual. Release numbers match the revision history in Release Information on page 2.

Table D-1: Issue 0000-01

Change	Location
First release	-