Arm® CoreLink™ GIC-625 Generic Interrupt Controller

Revision: r0p1

Technical Reference Manual

Non-Confidential

Issue 03
102143_0001_03_en

Arm® CoreLink™ GIC-625 Generic Interrupt Controller
Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	18 November 2020	Confidential	Limited access release for r0p0
0000-02	25 August 2021	Confidential	Early access release for r0p0
0001-03	15 December 2021	Non-Confidential	First release for r0p1

Proprietary Notice

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.

Previous issues of this document included language that can be offensive. We have replaced this language. See D Revisions on page 134.

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-625.

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.

Typographic conventions

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Signal names. 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 bold`	Language keywords when used outside example code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

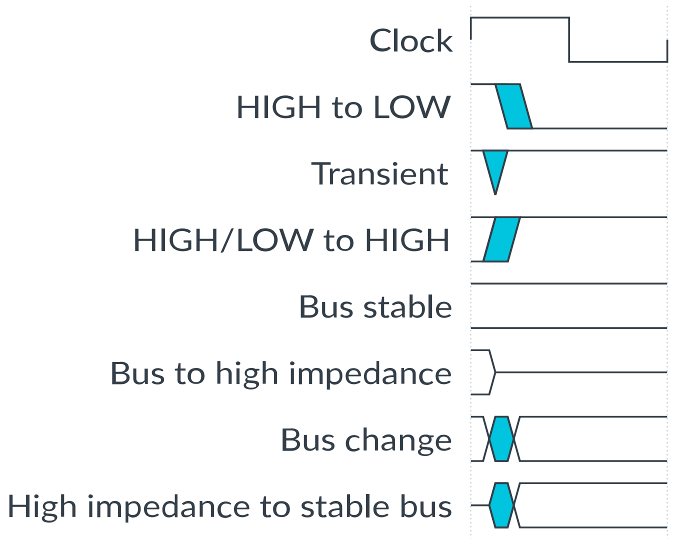
Convention	Use
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.
⚠ Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
⚠ Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
⚠ Danger	Requirements for the system. Not following these requirements will result in system failure or damage.
✎ Note	An important piece of information that needs your attention.
💡 Tip	A useful tip that might make it easier, better or faster to perform a task.
📌 Remember	A reminder of something important that relates to the information you are reading.

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 Additional reading

This document contains information that is specific to this product. See the following documents for other relevant information:

See *Developer* https://developer.arm.com/documentation/, for access to Arm documentation.

Confidential documents are available to licensees only, through the product bundle.

Table 1-2: Arm publications

Document name	Document ID	Confidentiality
Arm® CoreLink™ GIC-625 Generic Interrupt Controller Configuration and Integration Manual	102144	Confidential
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069G	Non-confidential

Document name	Document ID	Confidentiality
GICv3 and GICv4 Software Overview	DAI 0492	Non-confidential
AMBA® AXI and ACE Protocol Specification	IHI 0022G	Non-confidential
AMBA® AXI-Stream Protocol Specification	IHI 0051B	Non-confidential
AMBA® Low Power Interface Specification	IHI 0068D	Non-confidential
Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R AArch64 architecture profile	DDI 0600A.c	Non-confidential
Arm® Architecture Reference Manual Armv8, for A-profile architecture	DDI 0487G.b	Non-confidential
Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for Armv8-A	DDI 0587D.c	Non-confidential
Arm® CoreLink™ ADB-400 AMBA® Domain Bridge User Guide	DUI 0615	Confidential

Information published by third parties.

Table 1-3: Other publications

Document ID	Organization	Document name
JEP106	JEDEC	*Standard Manufacturer's Identification Code*

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-625

The GIC-625 is a generic interrupt controller that handles interrupts from peripherals to the cores, and interrupts between cores in a single cluster of up to 8 cores.

The GIC-625 supports the GICv3 and GICv3.1 architecture, see the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4.

2.1 Component overview

The GIC-625 comprises several significant blocks that work in combination to create a single architecturally compliant GICv3 and GICv3.1 implementation within the system.

The GIC-625 consists of the following blocks:

Distributor (GICD)

The Distributor is the hub of all the GIC communications and contains the functionality for all real-time *Shared Peripheral Interrupts* (SPIs). The Distributor supports up to 960 real-time SPIs, to use with devices that require a deterministic interrupt latency response. It is responsible for the entire GIC programmers model.

***GIC Cluster Interface* (GCI)**

The GCI maintains the *Private Peripheral Interrupts* (PPIs) and *Software Generated Interrupts* (SGIs) for a particular set of cores. A GCI can scale from 1-8 cores and is best placed next to the processors that it is servicing to reduce wiring to the cores.
A GCI is also referred to as a Redistributor.

The GICv3 architecture specifies a Redistributor address space containing two pages per core for GICv3. The SGI page functionality is contained in the GIC-625 Redistributor. However, the Distributor contains the other pages for all cores on a chip.

Wake Request

The Wake Request contains all the architecturally defined **wake_request** signals for each core on the chip.

The GIC-625 implements version 3 and 3.1 of the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4. To use GIC-625 with a core, the core must:

- Implement any of the Armv8.x-R architectures.
- Support the GIC Stream protocol interface.
- Support the extended range of GICv3.1 interrupts, when GIC-625 is configured and programmed to use >16 PPIs per core.

2.2 Compliance

The GIC-625 interfaces are compliant with Arm specifications and protocols.

The GIC-625 is compliant with:

- The AMBA® AXI5-Stream protocol. See the AMBA® AXI-Stream Protocol Specification.
- The AMBA® ACE5-Lite protocol. See the AMBA® AXI and ACE Protocol Specification.
- Version 3.1 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-625 provides interrupt services and masking, registers and programming, interrupt grouping, security, performance monitoring, and error correction.

Interrupt services and masking

The GIC-625 provides the following interrupt features:

- Support for the following interrupt types:
    - Up to 960 *Shared Peripheral Interrupts* (SPIs) in groups of 32 that can be assigned as real-time SPIs, to use with devices that require a deterministic interrupt latency response.
    - Up to 48 *Private Peripheral Interrupts*) (PPIs) that are independent for each core and can be programmed to support either edge-triggered or level-sensitive interrupts.
    - Up to 16 physical *Software Generated Interrupts* (SGIs) for each core, which the core generates through its GIC CPU interface.
- Interrupt masking and prioritization with 32 priority levels, 5 bits per interrupt.

Registers and programming

The GIC-625 provides the following programming features:

- Flexible affinity routing, using the *Multiprocessor Identification Register* (MPIDR) addresses, including support for four affinity levels (0-3).
- Single ACE5-Lite subordinate interface on each chip for programming of all registers.

Security

The GIC-625 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 28 for more information about security and groupings.

For more information about Exception levels, see the Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R AArch64 architecture profile.

Performance monitoring

The GIC-625 provides *Performance Monitoring Unit* (PMU) counters with snapshot functionality.

Error correction and containment

The GIC-625 provides the following error correction features:

- Armv8.2 *Reliability Accessibility Serviceability* (RAS) architecture-compliant error reporting for:
    - Software access 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.

2.4 Test features

The GIC-625 provides *Design for Test* (DFT) signals for test mode.

Related information

Common control signals on page 126

2.5 Product documentation

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

For relevant protocol and architectural information that relates to this product, see Additional reading.

The GIC-625 documentation is as follows:

Technical Reference Manual

The TRM describes the functionality and the effects of functional options on the behavior of the GIC-625. 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-625, contact:

- The implementer to determine:
  - The build configuration of the implementation
  - What integration, if any, was performed before implementing the GIC-625
- 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-625 into an SoC
- How to implement the GIC-625 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.

2.6 Product revisions

This section describes the differences in functionality between product revisions.

r0p0

First release

r0p0-r0p1

The functional changes are:

- Added the `spi_1ofn_support` configuration parameter, which can remove support for 1 of N interrupts. See 3.1.5 Distributor configuration on page 22 for more information.

3 Components and configuration

The GIC-625 contains several major components that use an internal GIC interconnect to route the AXI5-Stream interfaces between the different components.

The components are:

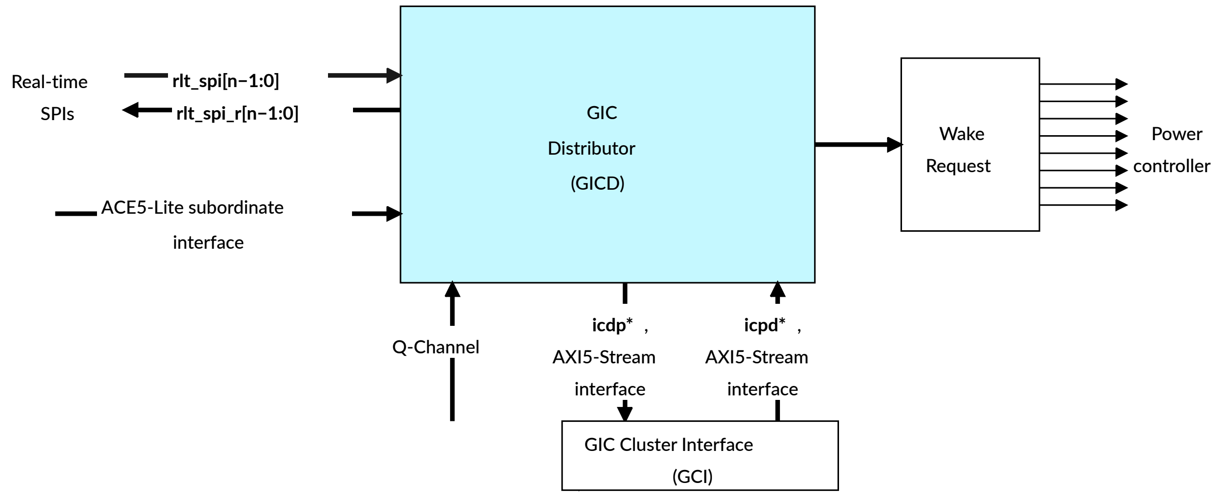
- Distributor
- *GIC Cluster Interface* (GCI)
- Wake Request

3.1 Distributor (GICD)

The Distributor is the main communication point between all GIC-625 blocks. It performs SPI management, and all communications with other blocks.

The following figure shows the Distributor and its interfaces.

Figure 3-1: GIC-625 Distributor



The Distributor is the main hub of the GIC and it implements most of the GICv3.1 architecture including:

- Programming, forwarding, and prioritization of SPIs, see 4.7 SPIs on page 34.
- SGI routing and forwarding, see 4.5 SGIs on page 32.
- Programming interface for all registers.
- Power control of cores.

3.1.1 Real-time SPI signals

The SPI inputs can be inverted and synchronized to the GICD **clk** when the appropriate parameters are set. The GICD also provides SPI outputs that can be used to create pulse extenders for edge-triggered interrupts that cross clock domains.

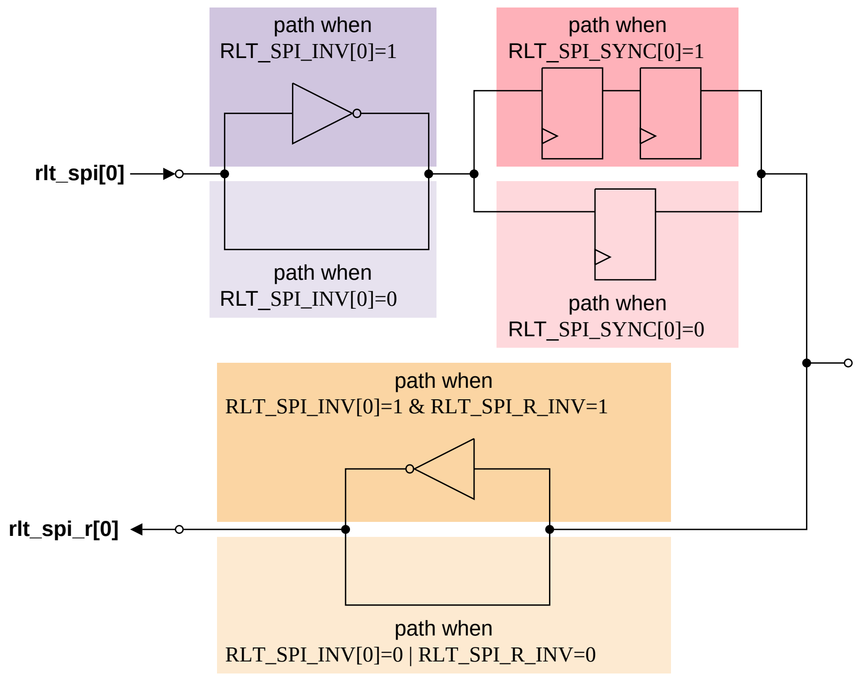
By default, the asserted level of an SPI is active-HIGH, as with previous Arm GIC implementations. However, each SPI can be either inverted, synchronized, or both, using the parameters `RLT_SPI_INV[n]` and `RLT_SPI_SYNC[n]`, where:

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

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

The following figure shows the effect of the `RLT_SPI_INV[n]`, `RLT_SPI_SYNC[n]`, and `RLT_SPI_R_INV` parameters on the **rlt_spi[0]** signal.

Figure 3-2: SPI parameters and signal conditioning



3.1.2 Distributor AXI5-Stream interfaces

The GIC-625 uses AXI5-Stream interfaces to communicate between blocks. These interfaces are internal and are not exposed to the system.

3.1.3 Distributor ACE5-Lite subordinate interface

The AMBA® ACE5-Lite subordinate port on the GIC-625 Distributor provides access to the entire register map. The interface supports 64-bit, 128-bit, 256-bit, or 512-bit data widths.

The GIC-625 only accepts single beat accesses of the sizes for each register that are shown in the programmers model, see 5 Programmers model on page 48.

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-625 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-625 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]** must all be tied LOW.

The GIC-625 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**) to ensure that the GIC does not request to be woken up due to incoming signal glitches.

The GIC-625 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 48.

Related information

Register map pages on page 48

3.1.3.1 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 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** returns SLVERR.
X	1	**rpoison** 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.

3.1.3.2 AMBA bus properties, GICD subordinate interface

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

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

Table 3-3: GICD ACE5-Lite subordinate interface properties

ACE5 property	Subordinate interface	ACE issue
Wakeup_Signals	TRUE	F
Check_Type	FALSE	F
Poison	TRUE	F
Trace_Signals	TRUE	F
Unique_ID_Support	TRUE	G

ACE5 property	Subordinate interface	ACE issue
QoS_Accept	FALSE	F
Loopback_Signals	TRUE	F
Untranslated_Transactions	FALSE	F
NSAccess_Identifiers	FALSE	F
CMO_On_Read	Ignore and respond legally	G
CMO_On_Write	FALSE	G
Persist_CMO	Ignore and respond legally	F
Write_Plus_CMO	FALSE	H
DVM_v8	FALSE	F
DVM_v8.1	FALSE	F
DVM_v8.4	FALSE	H
Coherency_Connection_Signals	FALSE	F
MPAM_Support	FALSE	G
Read_Interleaving_Disabled	No read data interleaving	G
Read_Data_Chunking	TRUE	G
Cache_Stash_Transactions	Support non-dataless, ignore, and respond legally – no stashing I/O.	F
Atomic_Transactions	Ignore and respond legally	F
DeAllocation_Transactions	Ignore and respond legally	F
WriteEvict_Transaction	TRUE	F
Barrier_Transactions	FALSE	F
MTE_Support	Basic	H
Prefetch_Transaction	FALSE	H
Fixed_Bursts	TRUE	F
Exclusive_Accesses	FALSE	F
Shareable_Transactions	TRUE	F
Max_Transaction_Bytes	TRUE	F

3.1.4 Distributor Q-Channel

There is a single Q-Channel for clock gating the GIC-625 Distributor, *GIC Cluster Interface* (GCI), and Wake Request components.

The **qreqn*** signals are synchronized internally, and can be driven asynchronously. See B.2 Power control signals on page 127.

As the **qactive** output 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 3-4: Configurable options for the Distributor

Feature	Range of options
Affinity0 width	0-4
Affinity1 width	0-8
Affinity2 width	0-8
Affinity3 width	0-4
Number of message-based SPIs permitted in system	32-960, in blocks of 32
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 130.
1 of N support	True, False

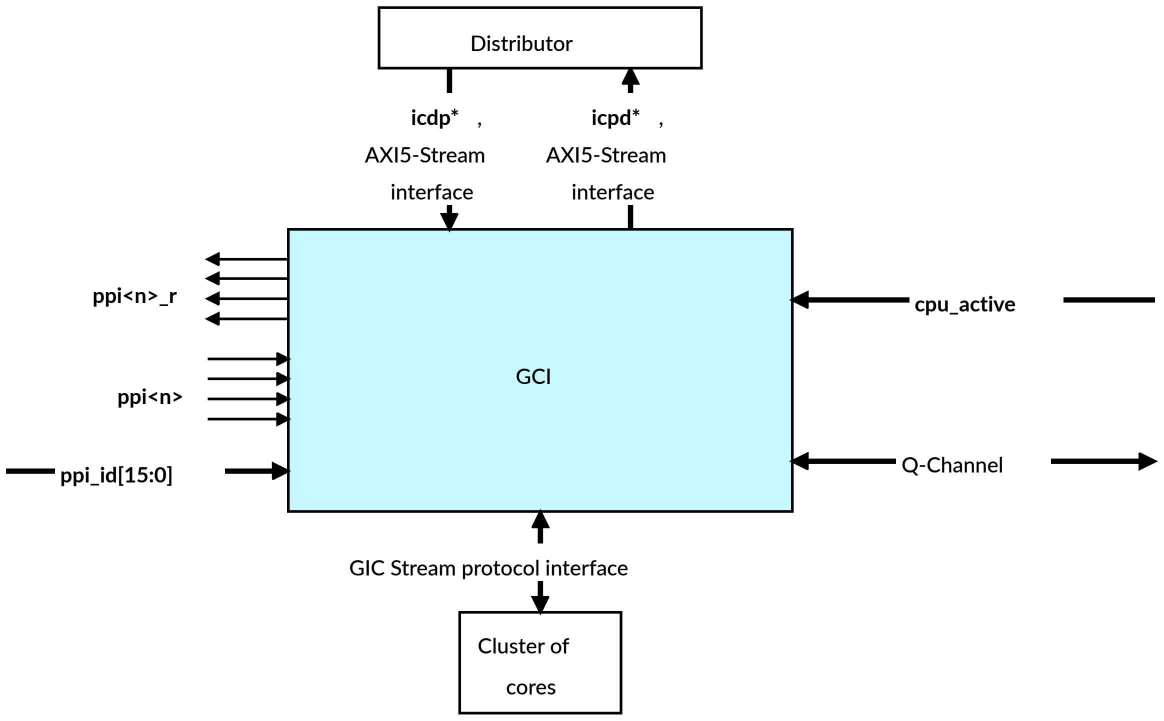
For more information, see the *Arm® CoreLink™ GIC-625 Generic Interrupt Controller Configuration and Integration Manual*.

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 3-3: GCI


Distributor
icdp* , AXI5-Stream interface
icpd* , AXI5-Stream interface
ppi<n>_r
ppi<n>
ppi_id[15:0]
GCI
cpu_active
Q-Channel
GIC Stream protocol interface
Cluster of cores

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-625 supports a single cluster of up to 8 cores.

The GCI (GICR) registers are programmed through the Distributor ACE5-Lite subordinate interface.

Related information

PPIs on page 33

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-625 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 AXI4-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 3-5: 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 requester. On this interface, the GCI is the requester and the CPU interface is the completer.
icc<*>	The **icc** prefix identifies the names of the upstream interface signals. These signals are sent by the GIC Stream completer. On this interface, the CPU interface is the requester and the GCI is the completer.
iritdest	The GCI uses this signal to direct packets to one core within the cluster.
icctid	The cluster uses this signal to determine which core within the cluster sent a packet.
iritwakeup	The GCI uses this signal to indicate that it wants to send a message to a CPU interface in the cluster.
icctwakeup	The cluster uses this signal to indicate that it wants to send a message to the GCI.

Both the **iritdest** and **icctid** can support 8 cores that use packed binary encoding, as opposed to one-hot encoding.

3.2.3 GCI PPI signals

GIC-625 supports 16, 32, or 48 PPIs, and synchronized output return wires, for each core. The number of PPIs and return wires must be the same for all cores that are sharing a GCI.

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

- `GIC625_<usrcfg>_PPI<ppi_id>_<cpu_number>_<ppi_number>_<INV>`
- `GIC625_<usrcfg>_PPI<ppi_id>_<cpu_number>_<ppi_number>_<SYNC>`
  Where `<usrcfg>` is user-defined text that is assigned when the GIC is configured, which can help with identifying a GIC configuration.

Every **ppi<n>** signal has a corresponding **ppi<n>_r** signal from after the synchronizer or capture flop. These **ppi<n>_r** signals can be used to create pulse extenders for edge-triggered interrupts that cross clock domains. The `GIC625_<usrcfg>_PPI<ppi_id>_<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 34

3.2.4 GCI configuration

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

Table 3-6: Configurable options for the GCI

Feature	Range of options
The number of cores that attach to this GCI	1-8
The number of PPIs for each core. To support more than 16 PPIs, the core must support the GICv3.1 extensions.	16, 32, 48

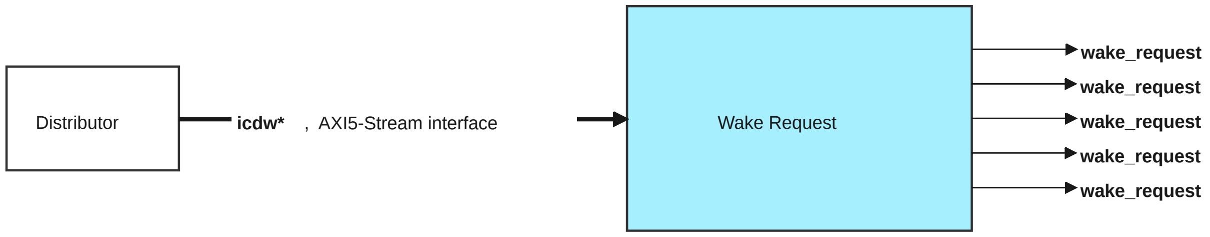
For more information, see the *Arm® CoreLink™ GIC-625 Generic Interrupt Controller Configuration and Integration Manual*.

3.3 Wake Request

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

The following figure shows the Wake Request block.

Figure 3-4: 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 has selected that core as its target.

The GIC-625 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, **wake_request** remains asserted after a core has powered up. **wake_request** deasserts when software clears GICR_WAKER.ProcessorSleep and the GIC-625 clears the GICR_WAKER.ChildrenAsleep bit.

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

The level of the asserted **wake_request[<cpus>−1:0]** signal drops only when the Distributor leaves reset, or when the core is woken and the GICR_WAKER.ProcessorSleep bit is cleared to indicate that it is able to communicate with the GIC. The GIC supports a Wake Request block reset only when the Distributor is also reset.

Related information

Power control signals on page 127

3.3.1 Wake Request configuration

The configuration of the Wake Request block is based on the number of cores in the system. There are no other options to configure.

3.4 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-625 file, `gic625_<usrcfg>.v`.

4 Operation

This chapter provides an operational description of the GIC-625 product.

4.1 Interrupt types

The GIC-625 manages SPIs, SGIs, and PPIs.

4.2 Interrupt groups and security

The GIC-625 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
- GICD_IGRPMODRn
- 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-625 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_ctrl_ds** tie-off signal controls the GIC-625 security as follows:

gicd_ctrl_ds is LOW

Security enabled

gicd_ctrl_ds is HIGH

Security disabled

Setting the **gicd_ctlr_ds** tie-off signal HIGH removes the security support of the GIC-625. Software can determine the state of this signal by reading the GICD_CTLR.DS bit. When the system has no concept of security, **gicd_ctlr_ds** must be set HIGH to allow access to important registers.

If **gicd_ctlr_ds** 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 **gicd_ctlr_ds** is only set HIGH when your system does not support security. See *Security model* in the GICv3 and GICv4 Software Overview for more information about the implications of disabling 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 **gicd_ctlr_ds** LOW.

For more information, see the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4 and the GICv3 and GICv4 Software Overview.

4.3 Affinity routing and assignment

The GIC-625 uses affinity routing, a hierarchical scheme, to identify connected cores and for routing interrupts to specific cores.

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

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

The GIC-625 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. 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-625 allows fully flexible allocation of MPIDR. However, it has two built-in default assignments that are based on the *`aff0_thread`* configuration parameter:

***`aff0_thread`* == 1**

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

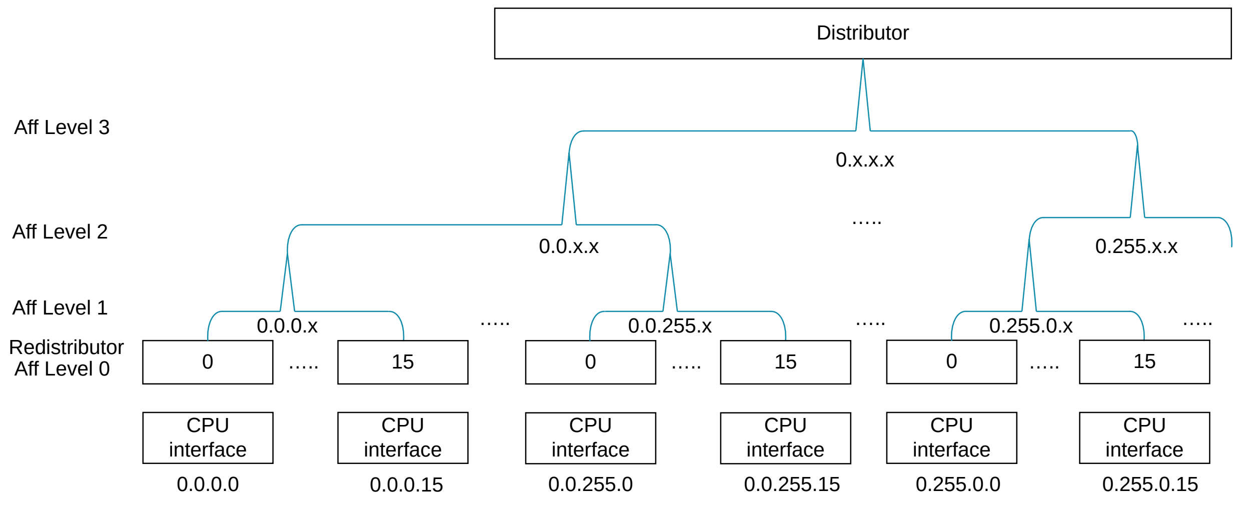
***`aff0_thread`* == 0**

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

See the *Arm® CoreLink™ GIC-625 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



For more information about affinity routing, see the GICv3 and GICv4 Software Overview and the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4.

4.4 RAMs and ECC

The GIC-625 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-625 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 4.10 Reliability, Accessibility, and Serviceability on page 41 for more information.

4.4.1 RAM error simulation

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

The GICD_ERRINSRn applies to the following RAMs:

- `0x0` = SGI RAM.
- `0x3` = SPI TGT RAM. This RAM is only present when *`spi_1ofn_support`* == 1.

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.

4.4.2 Scrub

The GIC-625 holds the interrupt states in RAM, which is protected by *Single Error Correction and Double Error Detection* (SECDED).

However, some RAM content 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 GICD_FCTLR.SIP bit. This process triggers a check and if necessary, a writeback of all valid RAM entries. Any errors that are found during a scrub are also reported in the relevant RAS error record.

4.5 SGIs

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

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

4.5.1 SGI programming

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

4.5.2 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 side.
4. If necessary, read out any of the current programmed states. This includes programmed data that is corrupted and generates an error, unless GICT_ERR0CTRL.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 32 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 32.

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 45

4.6 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 only sent 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-625 provides an option, through parameters, to include one or both a synchronizer and inverter on each PPI interrupt signal. See 3.2.3 GCI PPI signals on page 25 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.

4.6.1 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 28 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 GICv3 and GICv4 Software Overview and the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4.

4.6.2 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 84

4.7 SPIs

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

You can program each SPI to target either a particular core or any core. Activating an SPI on one core activates the SPI for all cores. That is, the GIC-625 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.

Real-time SPIs are generated by wire inputs. The GIC-625 can support up to 960 SPIs corresponding to the **rlt_spi** input signals on the Distributor. 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, from ID32 to ID991.

You can configure whether each SPI is triggered on a rising edge or is active-HIGH level-sensitive. The GIC-625 provides an option, through a parameter, to include one or both of a synchronizer or inverter for each SPI interrupt wire.

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.

4.7.1 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 real-time SPI signals are **rlt_spi**.

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 4.1 Interrupt types on page 28 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.

The GIC-625 provides optional synchronizers on every interrupt wire input. The GIC also providers return signals, **rlt_spi_r**, to enable the use of pulse extenders when sending edge-triggered interrupts across domain boundaries, see 3.1.1 Real-time SPI signals on page 18.

For more information, see the GICv3 and GICv4 Software Overview and the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4.

4.7.2 Low latency support

GIC-625 can be integrated into systems that require interrupts to be distributed to real-time peripherals with a deterministic low latency. To support this requirement, GIC-625 provides up to 960 real-time SPIs and up to 48 real-time PPIs.

The GIC-625 uses an inbuilt dedicated fast path to deliver the *Highest Priority Pending Interrupt* (HPPI) to a target core. The GIC assigns an interrupt as the HPPI when all the following conditions apply:

- the interrupt is pending, enabled, and not active
- the corresponding group enables are enabled in the GIC and processor
- the interrupt is the highest priority interrupt among all other valid interrupts to a particular core.

Note

- For SPIs with the same priority, the GIC considers the lower ID as higher in priority. Therefore, we recommend spreading out latency critical interrupt with the same priority across several cores, otherwise higher ID interrupts are not delivered until the lower ID interrupt is delivered and activated.
- For PPIs and SGIs, the GIC makes a random selection between available interrupts with the highest priority.

To achieve low latency for interrupts, we recommend that you set GICR_FCTLR.ECP to 1.

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

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

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

4.7.3 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 50

4.7.4 SPI routing and 1 of N selection

If GICD_TYPER.No1N==0, then the GIC-625 supports 1 of N selection of SPI interrupts. You can program an SPI to target several cores, and the GIC-625 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-625 only sends an SPI to cores that are powered up and have the relevant interrupt group enabled. The GIC-625 prioritizes cores that are considered active, but if there are no active cores, it selects inactive cores.

The selections that the GIC-625 makes can be controlled or influenced by several 1 of N features:

cpu_active

A **cpu_active** signal is an input to a Redistributor that corresponds to a particular core. When **cpu_active** 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** 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-625 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-625 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-625 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 **cpu_active** 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 Power management

The GIC-625 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.

4.8.1 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-625 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 powerup and powerdown sequences are 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;

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.

4.8.2 Processor core power management

The GIC architecture defines the programming sequence to safely power down a core that connects to the GIC-625.

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-625 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 3.3 Wake Request on page 26.

You must not set the GICR_WAKER.ProcessorSleep bit to 1 unless the core enters a power state where the GIC-625 uses the power controller to wake the core instead of the GIC Stream interface. For example, with Arm® Cortex®-A53 and Cortex®-A57 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 Armv8, for A-profile architecture. The system integrator can use **cpu_active** 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 **cpu_active** and are always sent to that core. Moreover, 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.

4.9 Performance Monitoring Unit

The GIC-625 contains a PMU for counting the main GIC events from the Distributor.

Note

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

The GIC events are described in Table 5-62: GICP_EVTYPERn.EVENT field encoding on page 109.

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 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 on a per counter basis by enabling 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 can be used for save and restore operations and for testing the correct integration of the **pmu_int** interrupt.

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

4.10 Reliability, Accessibility, and Serviceability

The GIC-625 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.

4.10.1 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.

4.10.2 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.
- 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.

4.10.3 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, so the interrupt only fires when the counter in the associated GICT_ERR<n>MISC0 register overflows. You can preset the counter to any value by writing to GICT_ERR<n>MISC0.Count. For example, to fire an interrupt on any correctable error, write `0xFF`, or to fire an interrupt on every second correctable error, write `0xFE`.

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

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

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

We recommend that if the **err_int** and **fault_int** are internally routed, the target interrupts must not have SPI 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** 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.

4.10.4 Error handling records

The GIC-625 has several error records. The range of error handling records that are available depends on the configuration of the GIC-625.

The following table lists the GIC-625 error handling records. The Type column uses the following acronyms:

CE	Correctable error
UEO	Restartable error and contained
UER	Recoverable error

Table 4-3: Error handling records

Record	Description	Type	Description, events, and recovery sequences
0	Software error in GICD programming	UEO	Table 4-4: Software errors, record 0 on page 44
3	Correctable SGI RAM errors	CE	Table 4-5: SGI RAM errors, records 3-4 on page 45. 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 4-6: TGT-SPI RAM errors, records 5-6 on page 46. GICT_ERR<n>STATUS.SERR == 7, control value from associative memory. These records are only present when GICD_TYPER.No1N==0.
6	Uncorrectable TGT-SPI cache errors	UER	

4.10.4.1 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-625. 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 4.10.3 Error recovery and fault handling interrupts on page 42 for more information.

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

Table 4-4: 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.
`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.
`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.
`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.
`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.

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
`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.
`0x1B`, SYN_SPI_NO_DEST_1OFN A 1 of N SPI cannot be delivered due to bad GICR_CTRL.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.
`0x1D`, SYN_DEACT_IN A Deactivate command to a nonexistent SPI, or a 1 of N SPI with incorrect groups set. Deactivate commands to 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. Incorrect groups are only reported for 1 of N SPIs.

4.10.4.2 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 4.10.3 Error recovery and fault handling interrupts on page 42 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 GICT_ERR<n>MISC0 reports data for SGI error records 3-4 shown in the following table.

Table 4-5: SGI RAM errors, records 3-4

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

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

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

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

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

Related information

SGI error recovery procedure on page 32

4.10.4.3 TGT-SPI RAM error records 5-6

The TGT-SPI RAM is only present when GICD_TYPER.No1N==0. 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.

The GICT_ERR<n>MISC0 register reports data for TGT-SPI error records 5-6 as the following table shows.

Table 4-6: 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.

4.10.4.4 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 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 32

4.10.5 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 4-7: 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

All the GIC-625 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-625 registers:

- The GIC-625 implements only memory-mapped registers.
- The GIC-625 has a single base address. 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-625 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-625 guarantees single-copy atomicity for doubleword accesses.

- The GIC-625 supports data only in little-endian format.
- The access types for the GIC-625 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 register map is separated into several pages.

The register map pages are defined in the following table.

Table 5-1: Register map pages

Offset[x:16]	Page	Description
0	GICD	GICD main page

Offset[x:16]	Page	Description
1	GICM	GICM message-based interrupts
2	GICT	GIC trace and debug page
3	GICP	GIC PMU page
4 + 2×RDnum	GICR (control)	GICR control registers
5 + 2×RDnum	GICR (PPI + SGI)	GICR PPI + SGI registers. RDnum is the serial number of each “internal Redistributor”, which is from 0 to RDcount−1.
4 + 2×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.

The GIC-625 ignores address bits above ceil[$\log_2$(page_count)] + 15. For example, a configuration that uses 11 pages ignores address bits above 19, and any address bits of the form `0xXXXXX00000` is accepted to access the GICD page of the memory map.

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

5.1.1 Discovery

We recommend that the operating system is provided with pointers to the start of the Distributor and the first Redistributor page on each chip.

To verify that the pages relate to GIC registers, software can check these pointers against the discovery registers, which start at offset `0xFFD0` for each GIC page. These registers allow discovery of the architecture version and, for GIC-625, whether the page contains the Distributor 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.

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 per Redistributor that are based on the features implemented. It can be inferred that GIC-625 has only two pages for each Redistributor because the GICR_TYPER.VLPIS bit indicates that it does not support 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.

For more information, see the GICv3 and GICv4 Software Overview.

5.1.2 GIC-625 register access and banking

The GIC-625 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-625 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	Configuration dependent	32	Distributor Implementer Identification Register	Yes
`0x000C`	GICD_TYPER2	RO	Configuration dependent	32	Interrupt Controller Type 2 Register	Yes

Offset	Name	Type	Reset	Width	Description	Architecture defined?
0x0010-0x001C	-	-	-	-	Reserved	-
0x0020	GICD_FCTLR	RW	0x0	32	Function Control Register	No
0x0024	GICD_SAC	RW	Tie-off dependent[1]	32	Secure Access Control register	No
0x0028-0x002C	-	-	-	-	Reserved	-
0x0030	GICD_FCTLR2	RW	0x0	32	Function Control Register 2	No
0x0034-0x003C	-	-	-	-	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[2][3]	WO	-	32	Secure SPI Set Register	Yes
0x0054	-	-	-	-	Reserved	-
0x0058	GICD_CLRSPI_SR[2][3]	WO	-	32	Secure SPI Clear Register	Yes
0x005C-0x007C	-	-	-	-	Reserved	-
0x0080-0x00FC	GICD_IGROUPR[3]	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 wire dependent	32	Interrupt Set-Pending Registers, n = 0-31, but n=0 is Reserved	Yes
0x0280-0x02FC	GICD_ICPENDRn	RW	SPI wire 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	Yes
0x0800-0x0BFC	-	-	-	-	Reserved	-
0x0C00-0x0CFC	GICD_ICFGRn	RW	0x0	32	Interrupt Configuration Registers, n = 0-61, but n=0-1 are Reserved	Yes

[1] The reset values of GICD_SAC.GICTNS and GICD_SAC.GICPNS are controlled by the **gict_allow_ns** and **gicp_allow_ns** tie-off signals respectively.
[2] The existence of this register depends on the configuration of the GIC-625. If Security support is not included, then this register is Reserved.
[3] This register is only accessible from a Secure access.

Offset	Name	Type	Reset	Width	Description	Architecture defined?
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[2]	RW	0x0	32	Non-secure Access Control Registers, n = 2-61	Yes
0x0F00-0x5FFC	-	-	-	-	Reserved	-
0x6000-0x7FF8	GICD_IROUTERn[4]	RW	0x0080000000 if configured.	64	Interrupt Routing Registers, n = 0-991, but n=0-31 are Reserved. See the GICv3 and GICv4 Software Overview. All SPIs are reset with Interrupt_Routing_Mode == 1. The first register is GICD_IROUTER32.	Yes
0x8000-0xDFFC	-	-	-	-	Reserved	-
0xE000-0xE0FC	GICD_ICLARn	RW	0x0	32	Interrupt Class Registers, n = 0-61, 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
0xE280-0xE9FC	-	-	-	-	Reserved	-
0xEA00-0xEA70	GICD_ERRINSRn	RW	Configuration dependent	64	Error Insertion Registers, n = 0-14	No
0xEA78-0xEFFC	-	-	-	-	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

[4] The first 32 of these registers do not exist when affinity routing is enabled.

Offset	Name	Type	Reset	Width	Description	Architecture defined?
`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 and affinity routing.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.2 Distributor registers (GICD/GICDA) summary on page 50 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

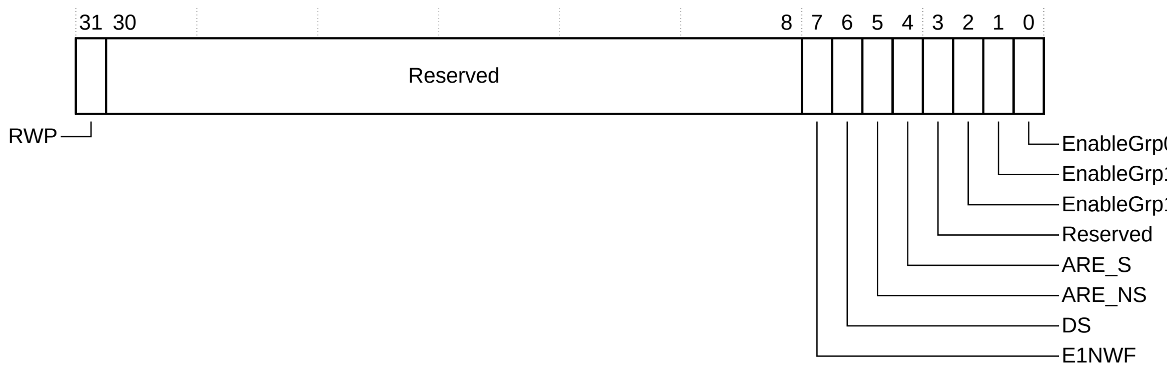


Table 5-3: GICD_CTLR bit descriptions

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	-	-

Bits	Name	Description	Type	Reset
[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.	RO	**gicd_ctlr_ds** signal
[5]	ARE_NS	Affinity Routing Enable, Non-secure state	RO	1
[4]	ARE_S	Affinity Routing Enable, Secure state	RO	1
[3]	-	Reserved	-	-
[2]	EnableGrp1S	Enable Secure Group 1 interrupts	RW	0
[1]	EnableGrp1NS	Enable Non-secure 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-625. 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 50 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-4: GICD_TYPER bit assignments

Bits	Name	Description
[31:27]	ESPI_Range	Returns zero, to indicate that the GIC-625 does not support any extended SPIs.
[26]	RSS	Range selector support. Returns: **0** The GIC supports targeted SGIs with affinity level 0 values of 0-15.

Bits	Name	Description
[25]	No1N	1 of N SPI: **0** The GIC-625 supports 1 of N SPI interrupts. This value occurs when *`spi_1ofn_support`* == 1. **1** The GIC-625 does not support 1 of N SPI interrupts. This value occurs when *`spi_1ofn_support`* == 0.
[24]	A3V	Affinity level 3 values. Depending on the configuration, returns either: **0** The GIC-625 Distributor only supports zero values of affinity level 3. **1** The GIC-625 Distributor supports nonzero values of affinity level 3.
[23:19]	IDbits	Interrupt identifier bits: **`0b01111`** The GIC-625 supports 16 interrupt identifier bits.
[18]	DVIS	Direct virtual LPI injection support: **0** The GIC-625 does not support direct virtual LPI injection. See the GICv3 and GICv4 Software Overview.
[17]	LPIS	Indicates whether the GIC supports LPIs: **0** The GIC-625 does not support LPIs.
[16]	MBIS	Message-based interrupt support: **1** The GIC-625 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** was HIGH during reset, so the GIC-625 supports only a single Security state. **1** **gicd_ctlr_ds** was LOW during reset, so the GIC-625 supports two Security states.
[9]	-	Reserved, RES0
[8]	ESPI	Extended SPI: **0** The GIC does not support the extended SPI range. Therefore, the GIC supports ≤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-625 implementation supports, and is given by 32×(ITLinesNumber + 1) − 1.

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

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-2: GICD_IIDR bit assignments

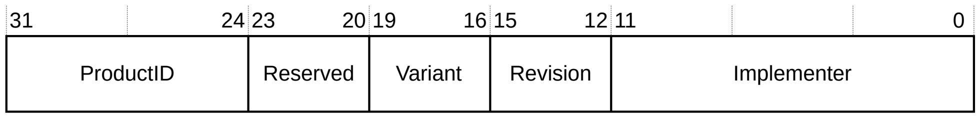


Table 5-5: GICD_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **`0x06`** GIC-625
[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 **`0x1`** p1
[11:0]	Implementer	Identifies the implementer: **`0x43B`** Arm

5.2.4 GICD_TYPER2, Interrupt Controller Type Register 2

This register returns the number of bits that GIC-625 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 50 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-3: GICD_TYPER2 bit assignments

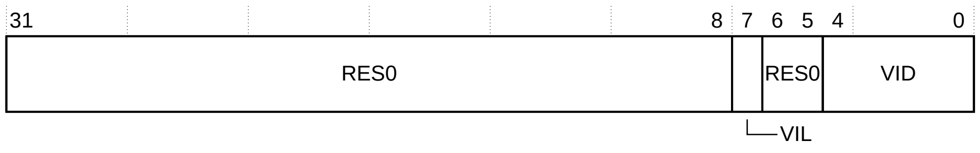


Table 5-6: GICD_TYPER2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RES0.
[7]	VIL	As GICD_TYPER.DVIS == 0, then this bit returns: **0** direct virtual LPI injection is not supported
[6:5]	-	Reserved, RES0.
[4:0]	VID	As GICD_TYPER.DVIS == 0, then this field returns: **0** direct virtual LPI injection is not supported

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

Usage constraints

Some bits are only accessible by Secure accesses.

Bit descriptions

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.

5.2.6 GICD_SAC, Secure Access Control register

This register allows Secure software to control Non-secure access to GIC-625 Secure features by other 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 50 for the address offset, type, and reset value of this register.

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Figure 5-4: GICD_SAC bit assignments

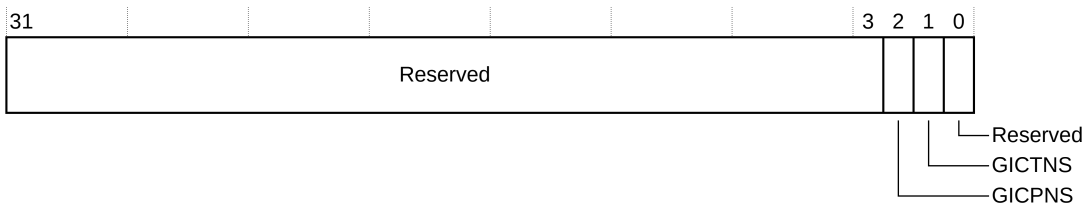


Table 5-8: GICD_SAC bit assignments

Bits	Name	Description	Type
[31:3]	-	Reserved, returns zero	-
[2]	GICPNS	Controls whether the Non-secure world can access the Secure PMU data: **0** Secure access only **1** Allow Non-secure access to the GICP registers The **gicp_allow_ns** tie-off signal controls the reset value on a per-chip basis.	RW
[1]	GICTNS	Controls whether the Non-secure world can access the Secure trace data: **0** Secure access only **1** Allow Non-secure access to the GICT registers The **gict_allow_ns** tie-off signal controls the reset value on a per-chip basis.	RW
[0]	-	Reserved, RES0	-

5.2.7 GICD_FCTLR2, Function Control Register 2

This register controls clock gating and other non-architectural controls in the local Distributor. The register is not distributed and only acts on the local chip.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.2 Distributor registers (GICD/GICDA) summary on page 50 for the address offset, type, and reset value of this register.

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

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
[27:19]	-	Reserved, RES0
[18]	QDENY	Q-Channel deny. Overrides the Q-Channel logic and forces the Distributor to reject powerdown requests.
[17:12]	-	Reserved, RES0

Bits	Name	Description
[11:0]	CGO	Clock gate override. One bit per clock gate: **0** Use full clock gating **1** Leave clock running. If clock gates are not implemented, then you must use this value. The clock gate bit assignments are: **Bit[11], CGO[11]** Real-time (RLT) block **Bit[10], CGO[10]** Reserved **Bit[9], CGO[9]** Reserved **Bit[8], CGO[8]** Reserved **Bit[7], CGO[7]** Reserved **Bit[6], CGO[6]** Trace and debug **Bit[5], CGO[5]** SGI and GICR registers **Bit[4], CGO[4]** Reserved **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

5.2.8 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-625 has 60 registers, GICD_ICLAR2-GICD_ICLAR61.

Configurations

This register is available in configurations when GICD_TYPER.No1N == 0.

Attributes

Width

32-bit

Functional group

See 5.2 Distributor registers (GICD/GICDA) summary on page 50 for the address offset, type, and reset value of this register.

Usage constraints

The Distributor provides up to 60 registers to support 960 SPIs. If you configure the GIC-625 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.

These registers are only accessible when the corresponding GICD_IROUTERn.Interrupt_Routing_Mode == 1.

Bit descriptions

Figure 5-5: GICD_ICLARn bit assignments

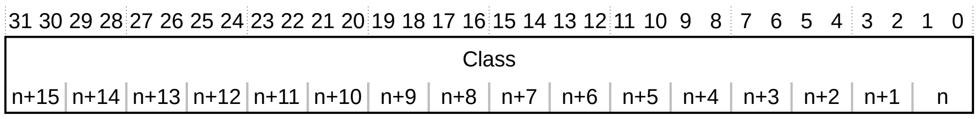


Table 5-10: GICD_ICLARn bit descriptions

Bits	Name	Description
[31:0] Bits[2`x`+1:2`x`], 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_ICLAR`n`, that is, SPI = 16×`n` + bit[number]/2.

5.2.9 GICD_ICERRRn, Interrupt Clear Error Registers

These registers are **RES0**. Each register monitors 32 SPIs and the GIC-625 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 50 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-625 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 5-6: GICD_ICERRRn bit assignments

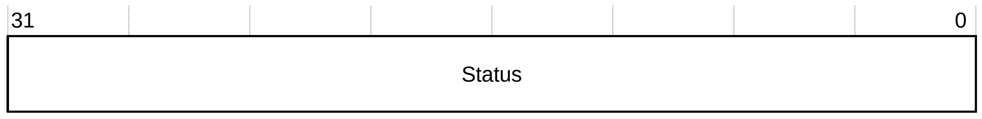


Table 5-11: GICD_ICERRRn bit descriptions

Bits	Name	Description
[31:0]	Status	**RES0**

5.2.10 GICD_ICGERRn, Interrupt Clear Group Error registers

These registers are **RES0**. Each register monitors 32 SPIs and the GIC-625 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 50 for the address offset, type, and reset value of this register.

Usage constraints

This register is Secure access only.

The Distributor provides up to 30 registers to support 960 SPIs. If you configure the GIC-625 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 5-7: GICD_ICGERRn bit assignments

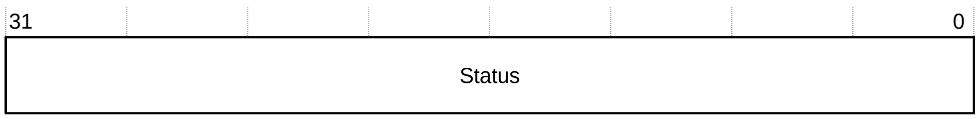


Table 5-12: GICD_ICGERRn bit descriptions

Bits	Name	Description
[31:0]	Status	**RES0**

5.2.11 GICD_ISERRRn, Interrupt Set Error Registers

These registers are **RES0**. Each register monitors 32 SPIs and the GIC-625 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 50 for the address offset, type, and reset value of this register.

Usage constraints

This register is Secure access only.

The Distributor provides up to 30 registers to support 960 SPIs. If you configure the GIC-625 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 5-8: GICD_ISERRRn bit assignments

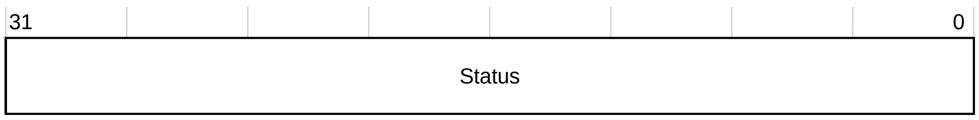


Table 5-13: GICD_ISERRRn bit descriptions

Bits	Name	Description
[31:0]	Status	**RES0**

5.2.12 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 50 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_SAC.GICTNS == 0, then only Secure software can access the functions of this register.

Bit descriptions

See 4.4.1 RAM error simulation on page 31 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 5-14: 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

Bits	Name	Description
[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 5-15: 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 only valid 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.

Related information

RAM error simulation on page 31

5.2.13 GICD_CFGID, Configuration ID Register

This register contains information that enables test software to determine if the GIC-625 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 50 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-16: 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.
[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
[31:21]	-	Reserved, returns zero
[20:15]	SPIS	Number of SPI blocks supported
[14]	-	Reserved
[13]	DLPI	Direct LPI registers supported **0** not supported
[12]	LPIS	LPI supported **0** not supported
[11:8]	ITSs	Returns zero because LPIS == 0 (no LPI support).
[7:4]	CNUM	Chip number
[3:1]	-	Reserved, returns zero
[0]	SO	Socket online status: **0** single chip

5.2.14 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 50 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-9: GICD_PIDR4 bit assignments

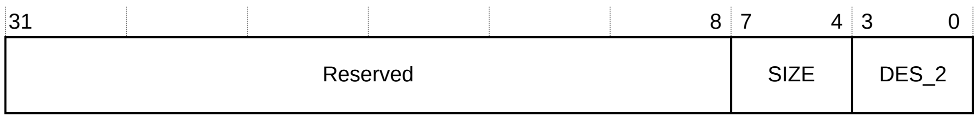


Table 5-17: GICD_PIDR4 bit descriptions

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.

5.2.15 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 50 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-10: GICD_PIDR3 bit assignments

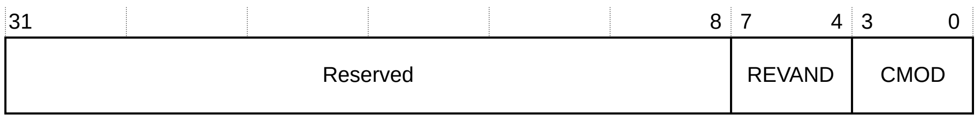


Table 5-18: GICD_PIDR3 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ.

Bits	Name	Description
[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`.

5.2.16 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 50 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-11: GICD_PIDR2 bit assignments

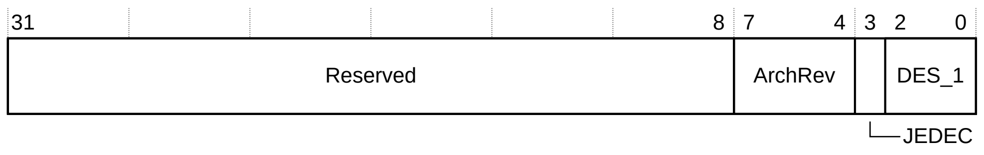


Table 5-19: GICD_PIDR2 bit descriptions

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.

5.2.17 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 50 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-12: GICD_PIDR1 bit assignments

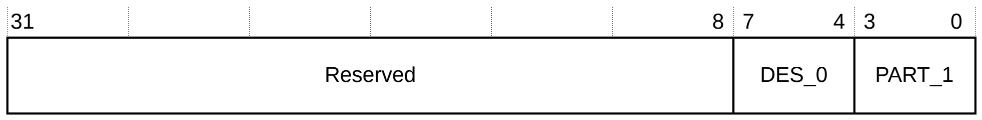


Table 5-20: GICD_PIDR1 bit descriptions

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.

5.2.18 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 50 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-13: GICD_PIDR0 bit assignments

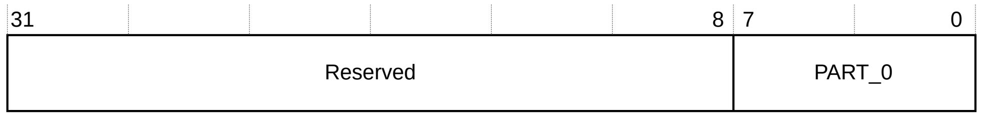


Table 5-21: GICD_PIDR0 bit descriptions

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-625 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 5-22: 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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
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[5]	WO[6]	-	32	Message-based Secure SPI Set Register	Yes
0x0054	-	-	-	-	Reserved	-
0x0058	GICM_CLRSPI_SR[5]	WO[6]	-	32	Message-based Secure SPI Clear Register	Yes
0x005C-0xFFC8	-	-	-	-	Reserved	-
0xFFCC	GICM_IIDR	RO	0x0300443B	32	Message-based Distributor Implementer Identification Register	Yes
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

[5] The existence of this register depends on the configuration of the GIC-625. If Security support is not included, this register does not exist.
[6] This register is only accessible from a Secure access.

Functional group

See 5.3 Distributor registers (GICM) for message-based SPIs summary on page 70 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-14: GICM_TYPER bit assignments

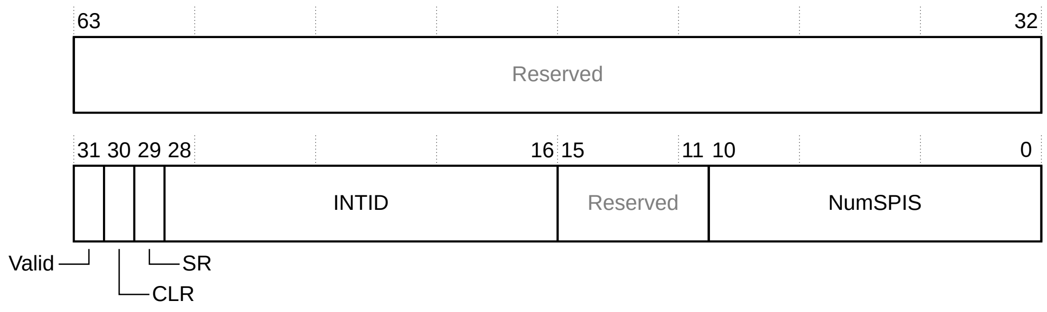


Table 5-23: GICM_TYPER bit descriptions

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. **Note:** 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 SPI IDs 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 70 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-15: GICM_IIDR bit assignments

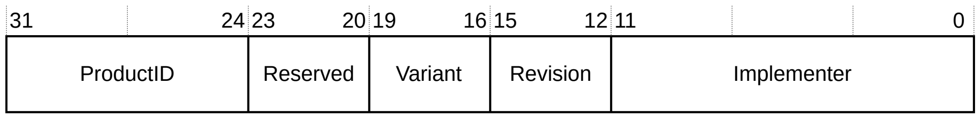


Table 5-24: GICM_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **`0x06`** GIC-625
[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 **`0x1`** p1
[11:0]	Implementer	Identifies the implementer: **`0x43B`** Arm

5.4 Redistributor control registers summary

The Redistributor functionality is controlled using the Redistributor registers that are identified with the prefix GICR. These registers start from the base address of the Redistributor.

For descriptions of registers that are not specific to the GIC-625, see the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4.

Table 5-25: Redistributor control registers summary

Offset	Name	Type	Reset	Width	Description	Architecture defined?
`0x0000`	GICR_CTLR	RW	`0x0`	32	Redistributor Control Register	Yes
`0x0004`	GICR_IIDR	RO	Configuration dependent	32	Redistributor Implementation Identification Register	Yes

Offset	Name	Type	Reset	Width	Description	Architecture defined?
`0x0008`	GICR_TYPER	RO	Configuration dependent	64	Redistributor Type Register	Yes
`0x0010`	-	-	-	-	Reserved	-
`0x0014`	GICR_WAKER	RW[7]	`0x6`	32	Power Management Control Register	[8]
`0x0018`	GICR_MPAMIDR	RO	`0x000101FF`	32	Report maximum PARTID and PMG Register	Yes
`0x001C`	GICR_PARTIDR	RW	`0x0`	32	Set PARTID and PMG Register	Yes
`0x0020`	GICR_FCTLR	RW	`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-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
`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.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

[7] This register is only accessible from a Secure access.
[8] Parts of this register are architecture defined and the other parts are microarchitecture defined.

Functional group

See 5.4 Redistributor control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-16: GICR_CTLR bit assignments

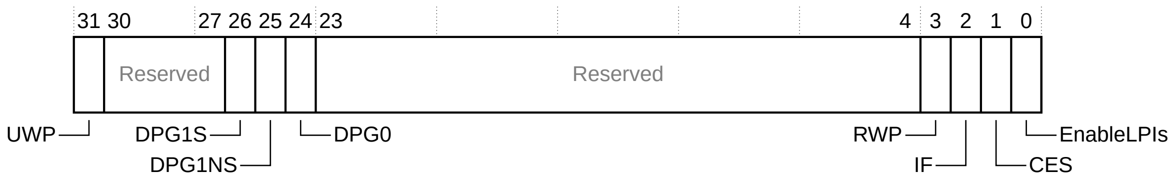


Table 5-26: GICR_CTLR bit descriptions

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. RAZ/WI when GICD_TYPER.No1N == 1.
[25]	DPG1NS	Disable processor selection for Group 1 Non-secure interrupts	
[24]	DPG0	Disable processor selection for Group 0 interrupts	
[23:4]	-	Reserved, RAZ	-
[3]	RWP	Register write pending: **0** No register write in progress **1** Register write in progress	RO
[2]	-	Reserved, RAZ	-
[1]	CES	Clear enable supported. Returns 1 to indicate that software can change GICR_CTLR.EnableLPIs from 1 to 0.	RO
[0]	EnableLPIs	RES0 because GIC-625 does not support physical LPIs.	-

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 control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-17: GICR_IIDR bit assignments

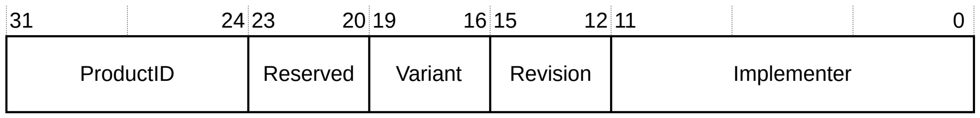


Table 5-27: GICR_IIDR bit descriptions

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID: **0x06** GIC-625
[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 **0x1** p1
[11:0]	Implementer	Identifies the implementer: **0x43B** Arm

5.4.3 GICR_TYPER, Redistributor Type Register

This register returns information about the features that this Redistributor supports.

Configurations

This register is available in all configurations.

Attributes

Width

64-bit

Functional group

See 5.4 Redistributor control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-28: GICR_TYPER bit descriptions

Bits	Name	Description
[63:32]	AffinityValue	Affinity level values for this Redistributor: **Bits[63:56], AF3** The affinity level 3 value **Bits[55:48], AF2** The affinity level 2 value **Bits[47:40], AF1** The affinity level 1 value **Bits[39:32], AF0** The affinity level 0 value
[31:27]	PPInum	Indicates the maximum PPI INTID that the GIC-625 supports: **0b00000** Maximum PPI INTID is 31 **0b00001** Maximum PPI INTID is 1087
[26]	-	Reserved, returns 0b0
[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]	-	Reserved, returns 0b0
[6]	MPAM	Returns 0 to indicate that *Memory Partitioning and Monitoring* (MPAM) is not supported.
[5]	DPGS	Indicates whether the GIC-625 supports *Disable Processor Group Selections*: **0** The GIC-625 does not support disable processor selections for Group 0 and Group 1 interrupts. **1** The GIC-625 supports disable processor selections for Group 0 and Group 1 interrupts. 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 this Redistributor does not support direct injection of LPIs. Therefore, the GICR_SETLPIR, GICR_CLRLPIR, GICR_INVLPIR, GICR_INVALLR, and GICR_SYNCR registers are not implemented.
[2]	Dirty	RES0 because GICR_TYPER.VLPIS == 0
[1]	VLPIS	Returns 0 to indicate that this Redistributor does not support virtual LPIs or the direct injection of virtual LPIs.
[0]	PLPIS	Returns 0 to indicate that this Redistributor does not support physical LPIs.

5.4.4 GICR_WAKER, Power Management Control Register

This register controls whether the GIC-625 can be powered down.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.4 Redistributor control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-18: GICR_WAKER bit assignments

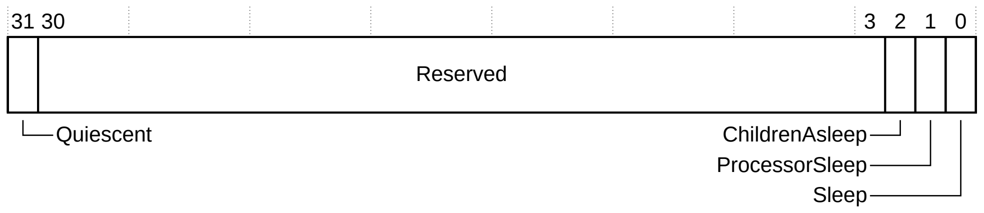


Table 5-29: GICR_WAKER bit descriptions

Bits	Name	Description
[31]	Quiescent	Indicates that the GIC-625 is idle and can be powered down if necessary
[30:3]	-	Reserved, RAZ
[2]	ChildrenAsleep	Indicates that the bus between the CPU interface and this Redistributor is quiescent
[1]	ProcessorSleep	Indicates: **0** This Redistributor never asserts **wake_request** and interrupt is delivered to the core **1** This Redistributor must assert a **wake_request** if there is a pending interrupt targeted at the connected core. See 4.8.2 Processor core power management on page 39.
[0]	Sleep	Indicates the sleep state: **0** Normal operation **1** The GIC-625 ensures that it is safe to power down. See A.1 Other power management on page 123.

5.4.5 GICR_MPAMIDR, Report maximum PARTID and PMG Register

This register is **RES0** because GICR_TYPER.MPAM == 0.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.4 Redistributor control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-19: GICR_MPAMIDR bit assignments

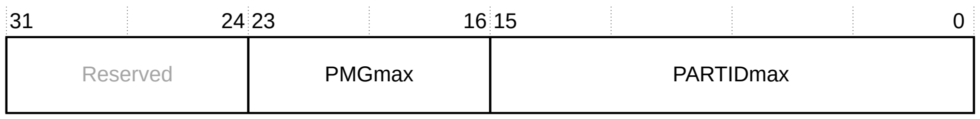


Table 5-30: GICR_MPAMIDR bit descriptions

Bits	Name	Description
[31:24]	-	Reserved.
[23:16]	PMGmax	**RES0**
[15:0]	PARTIDmax	**RES0**

5.4.6 GICR_PARTIDR, Set PARTID and PMG Register

This register is **RES0** because GICR_TYPER.MPAM == 0.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.4 Redistributor control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-20: GICR_PARTIDR bit assignments

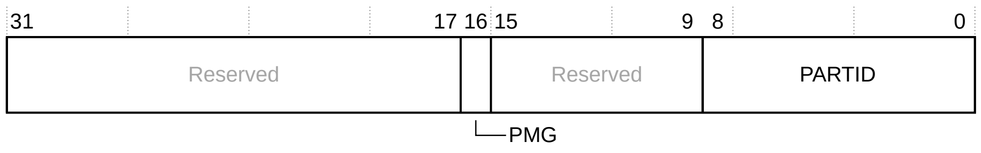


Table 5-31: GICR_PARTIDR bit descriptions

Bits	Name	Description
[31:17]	-	Reserved.
[16]	PMG	**RES0**
[15:9]	-	Reserved.
[8:0]	PARTID	**RES0**

5.4.7 GICR_FCTLR, Function Control Register

This register controls 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 control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-32: GICR_FCTLR bit descriptions

Bits	Name	Description
[31]	QD	Q-Channel deny: • 1 = Deny Q-Channel accesses • 0 = Allow Q-Channel accesses
[30:10]	-	Reserved, RAZ/WI

Bits	Name	Description
[9]	ECP	Enable combined packets. This bit controls whether the Redistributor combines packets to improve the latency when it connects to Arm® Cortex®-R82 cores: • 1 = The Redistributor combines GIC Stream messages, to improve the interrupt latency. • 0 = The Redistributor does not combine GIC Stream messages.
[8:7]	-	Reserved, RAZ/WI
[6:4]	CGO	Clock gate override. One bit per clock gate: • 1 = Leave clock running. If clock gates are not implemented, then you must use this value. • 0 = Use full clock gating The clock gate bit assignments are: **Bit[6], CGO[2]** Search clock gate **Bit[5], CGO[1]** Downstream message clock gate **Bit[4], CGO[0]** Upstream message clock gate
[3:1]	-	Reserved, RAZ/WI
[0]	SIP	Scrub in progress: • 1 = Scrub in progress • 0 = No scrub in progress This bit is read and written by software. When a scrub is complete, the GIC clears the bit to 0.

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 control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Figure 5-21: GICR_PWRR bit assignments

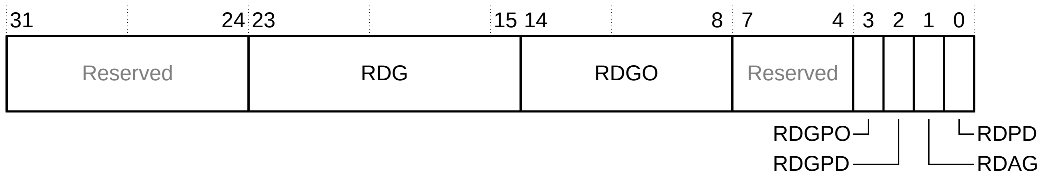


Table 5-33: GICR_PWRR bit descriptions

Bits	Name	Description	Type
[31:24]	-	Reserved, RAZ	-
[23:15]	RDG	RDGroup. This field indicates the number of the *GIC Cluster Interface* (GCI) of this Redistributor.	RO
[14:8]	RDGO	RDGroupOffset. This field indicates the identifier of the current core within the GCI.	RO
[7:4]	-	Reserved, RAZ	-
[3]	RDGPO	RDGroupPoweredOff. This bit indicates: • 0 = GCI is powered up and can be accessed. • 1 = It is safe to power down the GCI.	RO
[2]	RDGPD	RDGroupPowerDown. This bit indicates the intentional power state of the GCI: • 0 = Intend to power up • 1 = Intend to power down The GCI has reached its intentional power state when RDGPD = RDGPO.	RO
[1]	RDAG	RDApplyGroup. Setting this bit to 1 applies the RDPD value to all Redistributors on the same GCI. If the RDPD value cannot be applied to all cores in the group, then the GIC ignores this request.	WO
[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

Related information

Redistributor power management on page 38

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 configurations when GICD_TYPER.No1N == 0.

Attributes

Width

32-bit

Functional group

See 5.4 Redistributor control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Figure 5-22: GICR_CLASSR bit assignments

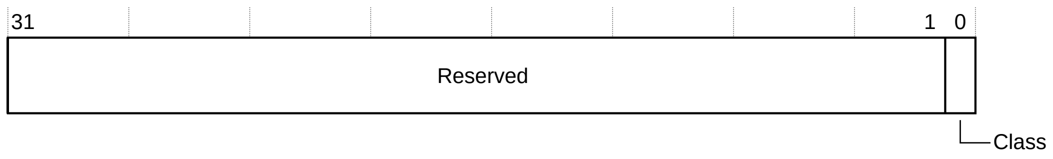


Table 5-34: GICR_CLASSR bit descriptions

Bits	Name	Description
[31:1]	-	Reserved, RAZ/WI.
[0]	Class	Interrupt class: **0** Class 0 **1** Class 1

Related information

SPI routing and 1 of N selection on page 36
GICD_ICLARn, Interrupt Class Registers on page 60

5.4.10 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 control registers summary on page 73 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-23: GICR_PIDR2 bit assignments

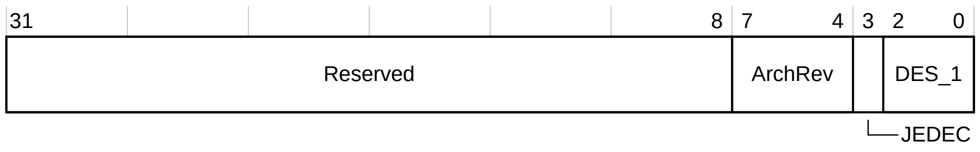


Table 5-35: GICR_PIDR2 bit descriptions

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.

5.5 Redistributor registers for SGIs and PPIs summary

The functions for the GIC-625 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-625, see the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4.

Table 5-36: 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
`0x0088-0x0FFC`	-	-	-	-	Reserved	-
`0x0100`	GICR_ISENABLER0	RW	`0x0`	32	Interrupt Set-Enable Register	Yes
`0x0104`	GICR_ISENABLER1E	RW	`0x0`	32	Interrupt Set-Enable Register Extended. Only present when *ppis_per_cpu* > 16.	Yes
`0x0108-0x017C`	-	-	-	-	Reserved	-
`0x0180`	GICR_ICENABLER0	RW	`0x0`	32	Interrupt Clear-Enable Register	Yes

Offset	Name	Type	Reset	Width	Description	Architecture defined?
`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 wire dependent	32	Interrupt Set-Pending Register	Yes
`0x0204`	GICR_ISPENDR1E	RW	`0x0`	32	Interrupt Set-Pending Register Extended. Only present when *`ppis_per_cpu`* > 16.	Yes
`0x0208-0x027C`	-	-	-	-	Reserved	-
`0x0280`	GICR_ICPENDR0	RW	PPI wire dependent	32	Peripheral Clear Pending Register	Yes
`0x0284`	GICR_ICPENDR1E	RW	`0x0`	32	Peripheral Clear-Pending Register Extended. Only present when *`ppis_per_cpu`* > 16.	Yes
`0x0288-0x02FC`	-	-	-	-	Reserved	-
`0x0300`	GICR_ISACTIVER0	RW	`0x0`	32	Interrupt Set-Active Register	Yes
`0x0304`	GICR_ISACTIVER1E	RW	`0x0`	32	Interrupt Set-Active Register Extended. Only present when *`ppis_per_cpu`* > 16.	Yes
`0x0308-0x037C`	-	-	-	-	Reserved	-
`0x0380`	GICR_ICACTIVER0	RW	`0x0`	32	Interrupt Clear-Active Register	Yes
`0x0384`	GICR_ICACTIVER1E	RW	`0x0`	32	Interrupt Clear-Active Register Extended. Only present when *`ppis_per_cpu`* > 16.	Yes
`0x0388-0x03FC`	-	-	-	-	Reserved	-
`0x0400-0x041C`	GICR_IPRIORITYRn	RW	`0x0`	32	Interrupt Priority Registers	Yes
`0x0420`	GICR_IPRIORITYRnE	RW	`0x0`	32	Interrupt Priority Registers Extended. Only present when *`ppis_per_cpu`* > 16.	Yes
`0x0440-0x0BFC`	-	-	-	-	Reserved	-
`0x0C00-0x0C04`	GICR_ICFGRn	RW	`0xAAAAAAAA` when `n` == 0. `0x0` when `n` == 1.	32	Interrupt Configuration Registers	Yes
`0x0C08-0x0C0C`	GICR_ICFGRnE	RW	`0x0`	32	Interrupt Configuration Registers Extended. Only present when *`ppis_per_cpu`* > 16.	Yes
`0x0C10-0x0CFC`	-	-	-	-	Reserved	-
`0x0D00`	GICR_IGRPMODR0	RW	`0x0`	32	Interrupt Group Modifier Register	Yes
`0x0D04-0x0C0C`	GICR_IGRPMODR1E	RW	`0x0`	32	Interrupt Group Modifier Register Extended. Only present when *`ppis_per_cpu`* > 16.	Yes
`0x0D08-0x0DFC`	-	-	-	-	Reserved	-
`0x0E00`	GICR_NSACR	RW	`0x0`	32	Non-secure Access Control Register	Yes

Offset	Name	Type	Reset	Width	Description	Architecture defined?
`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 as seen by the GIC-625.

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

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-24: GICR_MISCSTATUSR bit assignments

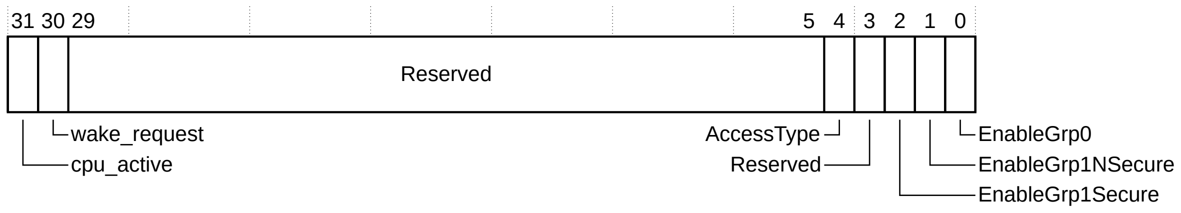


Table 5-37: GICR_MISCSTATUSR bit descriptions

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 not active • 1 = **cpu_active** input signal 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** not active • 1 = **wake_request** asserted
[29:5]	-	Reserved
[4]	AccessType	Returns the access type: • 0 = Secure access • 1 = Non-secure access
[3]	-	Reserved
[2][9]	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.

[9] These bits are a copy of the CPU interface group enables for the core corresponding to this Redistributor. These copies are undefined when ProcessorSleep or ChildrenSleep 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.

Bits	Name	Description
[1][9]	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][9]	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 84 for the address offset, type, and reset value of this register.

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Figure 5-25: GICR_ICDERRR bit assignments

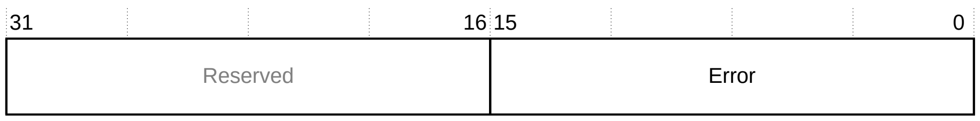


Table 5-38: GICR_ICDERRR bit descriptions

Bits	Name	Description
[31:16]	-	Reserved.
[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.

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

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Table 5-39: 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>.

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

Usage constraints

Some fields are only writable by using a Secure access.

Bit descriptions

Figure 5-26: GICR_DPRIR bit assignments

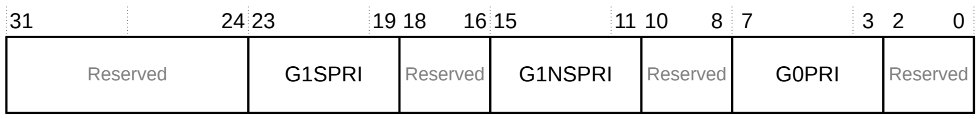


Table 5-40: GICR_DPRIR bit descriptions

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.

5.5.5 GICR_ICERRR0, Interrupt Clear Error Register 0

This register is **RES0.**

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

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Figure 5-27: GICR_ICERRR0 bit assignments

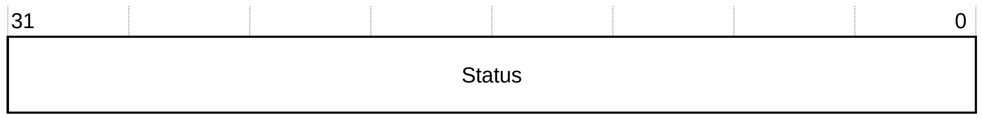


Table 5-41: GICR_ICERRR0 bit descriptions

Bits	Name	Description
[31:0]	Status	**RES0**

5.5.6 GICR_ICERRR1E, Interrupt Clear Error Register Extended

This register is reserved.

Configurations

This register is not present in configurations with 16 PPIs (when GICR_TYPER.PPInum == 0), and is reserved in all other configurations.

Attributes

Width

32-bit

Functional group

See 5.5 Redistributor registers for SGIs and PPIs summary on page 84 for the address offset, type, and reset value of this register.

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Figure 5-28: GICR_ICERRR1E bit assignments

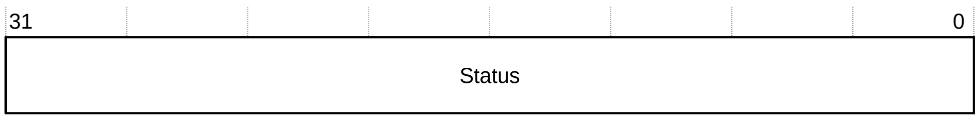


Table 5-42: GICR_ICERRR1E bit descriptions

Bits	Name	Description
[31:0]	status	Reserved

5.5.7 GICR_ISERRR0, Interrupt Set Error Register 0

This register is **RES0**.

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

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Figure 5-29: GICR_ISERRR0 bit assignments

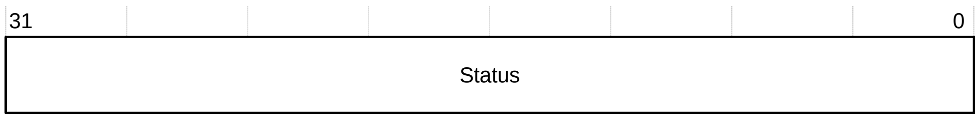


Table 5-43: GICR_ISERRR0 bit descriptions

Bits	Name	Description
[31:0]	Status	**RES0**

5.5.8 GICR_ISERRR1E, Interrupt Set Error Register Extended

This register is reserved.

Configurations

This register is not present in configurations with 16 PPIs (when GICR_TYPER.PPInum == 0), and is reserved in all other configurations.

Attributes

Width

32-bit

Functional group

See 5.5 Redistributor registers for SGIs and PPIs summary on page 84 for the address offset, type, and reset value of this register.

Usage constraints

Only accessible by Secure accesses.

Bit descriptions

Figure 5-30: GICR_ISERRR1E bit assignments

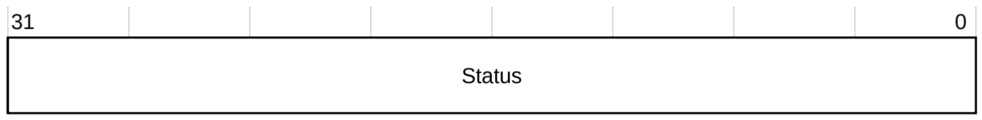


Table 5-44: GICR_ISERRR1E bit descriptions

Bits	Name	Description
[31:0]	Status	**RES0**

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

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-45: GICR_CFGID0 bit descriptions

Bits	Name	Description
[31:12]	-	Reserved, RAZ
[11]	ECCSupport	0 = ECC is not supported
[10:9]	-	Reserved, RAZ
[8:0]	PPINumber	RedistributorID. The **ppi_id[15:0]** tie-off signal sets the value of the ID. Each Redistributor must have a unique ID.

Related information

Miscellaneous signals on page 130

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

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-46: GICR_CFGID1 bit descriptions

Bits	Name	Description
[31:28]	Version	Identifies the major and minor revisions of GIC-625: **`0x0`** r0p0 **`0x2`** r0p1
[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.
[23:20]	-	Reserved, RAZ
[19:16]	PPIs_per_Processor	The number of PPIs for each core − 1

Bits	Name	Description
[15:12]	-	Reserved
[11:4]	NumCPUs	The number of cores that this Redistributor supports. GIC-625 supports up to 32 cores, so the maximum value of this field is `0x1F`.
[3:0]	-	Reserved, RAZ

5.5.11 GICR_ERRINSR, Error Insertion Registers

This register is **RES0**.

Configurations

This register is reserved in all configurations because the *GIC Cluster Interface* (GCI) has no RAM.

Attributes

Width

64-bit

Functional group

See 5.5 Redistributor registers for SGIs and PPIs summary on page 84 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_SAC.GICTNS == 0, then only Secure software can access the contents of this register.

Bit descriptions

Table 5-47: GICR_ERRINSR bit descriptions

Bits	Name	Description
[63:0]	-	**RES0**

Related information

RAM error simulation on page 31

5.6 GICT register summary

The GIC-625 trace and debug functions are controlled through registers that are identified with the prefix GICT.

All registers comply with the Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for Armv8-A, 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 5-48: GICT register summary

Offset	Name	Type	Reset	Width	Description
0x0000 + (n × 64)	GICT_ERR<n>FR	RO	Record dependent	64	Error Record Feature Register
0x0008 + (n × 64)	GICT_ERR<n>CTLR	RW	0x0	64	Error Record Control Register
0x0010 + (n × 64)	GICT_ERR<n>STATUS	RW	Record dependent	64	Error Record Primary Status register
0x0018 + (n × 64)	GICT_ERR<n>ADDR	RW	Unknown	64	Error Record Address Register
0x0020 + (n × 64)	GICT_ERR<n>MISC0	RW	Unknown	64	Error Record Miscellaneous Register 0
0xE000	GICT_ERRGSR	RO	0x0	64	Error Group Status Register
0xE008- 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- 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

5.6.1 GICT_ERR<n>FR, Error Record Feature Register

This register returns information about the Armv8.2 RAS features that the GIC-625 implements.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_SAC.GICTNS == 0, then only Secure software can access the contents of this register.

Bit descriptions

Figure 5-31: GICT_ERR<n>FR bit assignments

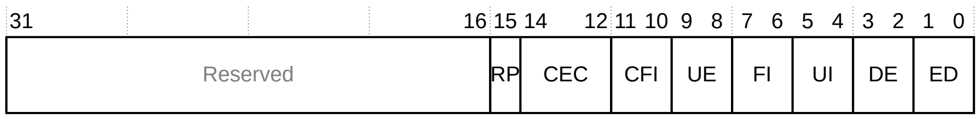


Table 5-49: GICT_ERR<n>FR bit descriptions

Bits	Name	Description
[31:16]	-	Reserved, RAZ.
[15]	RP	Repeat corrected error count: • 0 = The GIC-625 does not implement a repeat corrected error counter.
[14:12]	CEC	Corrected error count: • `0b000` = The GIC-625 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-625 does not provide a fault handling interrupt for corrected errors. • `0b10` = The GIC-625 provides a controllable fault handling interrupt for corrected errors.
[9:8]	UE	Uncorrected error. Depending on the configuration, returns either: • `0b00` = The GIC-625 does not provide an in-band uncorrected error reporting. • `0b10` = The GIC-625 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-625 does not provide a fault handling interrupt. • `0b10` = The GIC-625 provides a controllable fault handling interrupt.
[5:4]	UI	Error recovery interrupt for uncorrected errors. Depending on the configuration, returns either: • `0b00` = The GIC-625 does not provide an error recovery interrupt for uncorrected errors. • `0b10` = The GIC-625 provides a controllable error recovery interrupt for uncorrected errors.
[3:2]	DE	Deferring of errors support: • `0b00` = The GIC-625 does not support the deferring of errors.
[1:0]	ED	Uncorrected error reporting: • `0b01` = Uncorrected error reporting is always enabled.

5.6.2 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 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_SAC.GICTNS == 0, then only Secure software can access the functions of this register.

Bit descriptions

Table 5-50: 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: • 1 = Reporting of illegal ACE accesses is disabled • 0 = Illegal ACE accesses are treated as errors, which generate the SYN_ACE_BAD syndrome.
[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: • 1 = Reporting of out-of-range SGI destinations is disabled • 0 = Out-of-range SGI destinations are treated as errors, which generate the SYN_SGI_NO_TGT syndrome.
[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: • 1 = Reporting of SPIs with no available destination is disabled • 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.
[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: • `0b10` = Reporting of SPI register accesses to SPIs 992-1023 is disabled • `0b01` = Reporting of SPI register accesses to all nonexisting blocks is disabled • `0b00` = SPI register accesses to nonexisting blocks are treated as errors, which generate either a SYN_SPI_BLOCK or SYN_SPI_OOR syndrome.
[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: • 1 = Reporting of out-of-range deactivate messages is disabled • 0 = Out-of-range deactivate messages are treated as errors, which generate the SYN_DEACT_IN syndrome.

Bits	Name	Description
[32:16]	-	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-625 does not assert a fault handling interrupt for corrected errors. • 1 = The GIC-625 asserts a fault handling interrupt, **fault_int**, 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 4.10.5 Bus errors on page 47.
[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**, 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**, is generated on all uncorrectable errors
[1:0]	-	Reserved, RAZ

5.6.3 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 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_SAC.GICTNS == 0, then only Secure software can access the functions of this register.

Bit descriptions

Figure 5-32: GICT_ERR<n>STATUS bit assignments

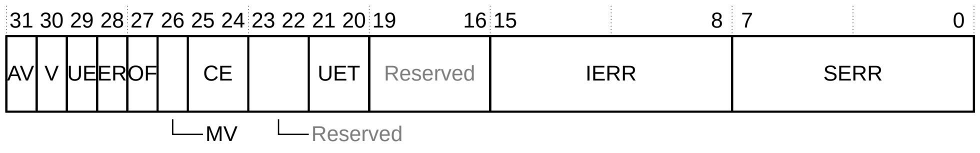


Table 5-51: GICT_ERR<n>STATUS bit descriptions

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. Only present 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 only for accesses to corrupted data, and bad incoming access.
[27]	OF	Record has overflowed
[26]	MV	Indicates if the GICT miscellaneous register is valid: • 0 = GICT_ERR<n>MISC0 is not valid • 1 = GICT_ERR<n>MISC0 is valid
[25:24]	CE	Correctable error. Indicates errors that are correctable as shown in Table 4-3: Error handling records on page 43: • `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 5-54: GICT_ERR<n>MISC0.Data field encoding on page 103 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 5-54: GICT_ERR<n>MISC0.Data field encoding on page 103 shows. See the Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for Armv8-A for more information about this field. This field is RO apart from record 0.

5.6.4 GICT_ERR<n>ADDR, Error Record Address Register

This register contains the address and security status of the write. This register is only present for GICT software record 0.

Configurations

This register is available in all configurations.

Attributes

Width

64-bit

Functional group

See 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_SAC.GICTNS == 0, then only Secure software can access the functions of this register.

Ignores writes if GICT_ERR<n>STATUS.AV == 1.

All bits are RAZ/WI if GICT_ERR<n>STATUS.IERR = 0, 12, or 13.

Bit descriptions

Table 5-52: 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].

5.6.5 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 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

None

Bit descriptions

Figure 5-33: GICT_ERR<n>MISC0 bit assignments

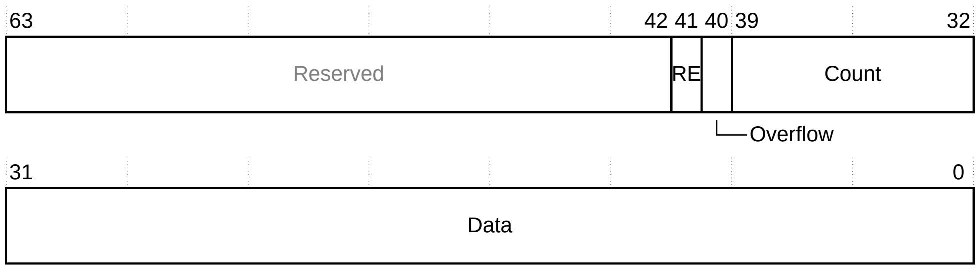


Table 5-53: GICT_ERR<n>MISC0 bit descriptions

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-625 generates a fault handling interrupt.
[39:32]	Count	Error count. Error counter is not 0 or is more than 27+. Incremented for each corrected error or uncorrectable error that does not match the recorded syndrome.
[31:0]	Data	Information that is associated with the error. A description of each error code is given in one of the following tables: • Table 4-4: Software errors, record 0 on page 44 • Table 4-5: SGI RAM errors, records 3-4 on page 45 • Table 4-6: TGT-SPI RAM errors, records 5-6 on page 46

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

Table 5-54: 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)	`0x7`, SYN_WAKER_CHANGE Attempt to change GICR_WAKER abandoned due to handshake rules.	`0xF`	Core, bits[8:0]
Software error (0)	`0x8`, SYN_SLEEP_FAIL Attempt to put GIC to sleep failed because cores are not fully asleep.	`0xF`	Core, bits[8:0]
Software error (0)	`0x9`, SYN_PGE_ON_QUIESCE Core put to sleep before its Group enables were cleared.	`0xF`	Core, bits[8:0]
Software error (0)	`0x10`, SYN_SGI_NO_TGT SGI sent with no valid destinations.	`0xE`	Core, bits[8:0]
Software error (0)	`0x12`, SYN_GICR_CORRUPTED Data was read from GICR register space that encountered an uncorrectable error.	`0x6`	GICT_ERR0ADDR is populated
Software error (0)	`0x18`, SYN_SPI_BLOCK. Attempt to access an SPI block that is not implemented.	`0xE`	Block, bits[4:0]
Software error (0)	`0x19`, SYN_SPI_OOR Attempt to access a non-implemented SPI using (SET|CLR)SPI.	`0xE`	ID, bits[9:0]
Software error (0)	`0x1B`, SYN_SPI_NO_DEST_1OFN A 1 of N SPI cannot be delivered due to bad GICR_CTLR.DPG<0|1NS|1S> or GICR_CLASSR programming.	`0xF`	ID, bits[9:0]
Software error (0)	`0x1D`, SYN_DEACT_IN A Deactivate command to a nonexistent SPI, or a 1 of N SPI with incorrect groups set. Deactivate commands nonexistent PPI are not reported.	`0xE`	None
Correctable SGI RAM errors (3)	**`0x00`** a real error **`0x01`** an injected error	`0x7`	See Table 4-5: SGI RAM errors, records 3-4 on page 45
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 4-6: TGT-SPI RAM errors, records 5-6 on page 46 The TGT-SPI RAM is only present when GICD_TYPER.No1N==0.
Uncorrectable TGT-SPI cache errors (6)	**`0x00`** a real error **`0x01`** an injected error	`0x7`	

5.6.6 GICT_ERRGSR, Error Group Status Register

This register shows the status of the GIC-625 Armv8.2 RAS architecture-compliant error records for correctable and uncorrectable RAM ECC errors, and uncorrectable software errors.

Configurations

This register is available in all configurations.

Attributes

Width

64-bit

Functional group

See 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-34: GICT_ERRGSR bit assignments

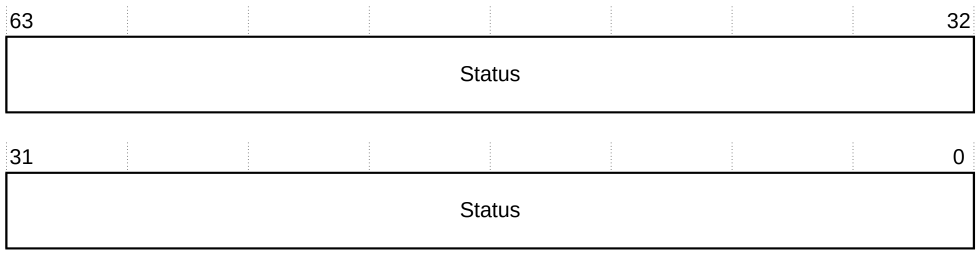


Table 5-55: GICT_ERRGSR bit descriptions

Bits	Name	Description
[n]	Status	Indicates the status of error record n, where n is 0-8 depending on the configuration: • 0 = The error record is not reporting any errors • 1 = The error record is reporting one or more errors

5.6.7 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 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_SAC.GICTNS == 0, then only Secure software can access the functions of this register.

Bit descriptions

Table 5-56: GICT_ERRIRQCR<n> bit descriptions

Bits	Name	Description
[31:11]	-	Reserved, RAZ
[10:0]	SPIID	SPI ID. Returns 0 if an invalid entry is written. **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

5.6.8 GICT_DEVID, Device Configuration register

This register returns information about the configuration of the GIC-625 GICT.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

If GICD_SAC.GICTNS == 0, then only Secure software can read this register.

Bit descriptions

Figure 5-35: GICT_DEVID bit assignments

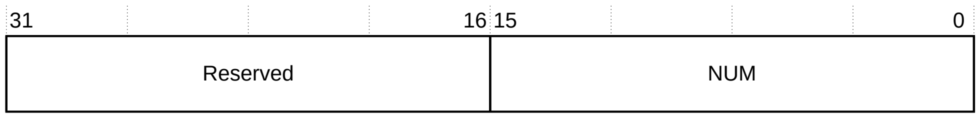


Table 5-57: GICT_DEVID bit descriptions

Bits	Name	Description
[31:16]	-	Reserved, RAZ
[15:0]	NUM	Identifies the device configuration: **10** No LPI available

5.6.9 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 5.6 GICT register summary on page 95 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-36: GICT_PIDR2 bit assignments

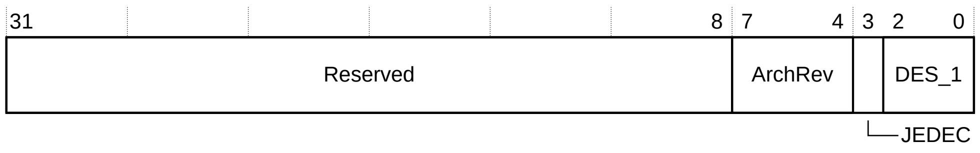


Table 5-58: GICT_PIDR2 bit descriptions

Bits	Name	Description
[31:8]	-	Reserved, RAZ

Bits	Name	Description
[7:4]	ArchRev	Identifies the version of the GIC architecture with which the trace and debug block complies: • `0x3` = GICv3 • `0x4` = GICv4
[3]	JEDEC	Indicates that a JEDEC-assigned JEP106 identity code is used.
[2:0]	DES_1	Bits[6:4] of the JEP106 identity code. Bits[3:0] of the JEP106 identity code are assigned to GICT_PIDR1.

5.7 GICP register summary

The GIC-625 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 5-59: 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
`0xE50`	GICP_IRQCR	RW	`0x0`	32	Interrupt Configuration Register	No
`0xFB8`	GICP_PMAUTHSTATUS	RO	`0x088`	32	-	-
`0xFBC`	GICP_PMDEVARCH	RO	`0x47702A56`	32	-	-
`0xFCC`	GICP_PMDEVTYPE	RO	`0x56`	32	-	-
`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

Offset	Name	Type	Reset	Width	Description	Architecture defined?
`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

5.7.1 GICP_EVCNTRn, Event Counter Registers

These registers contain the values of event counter `n`. The GIC-625 supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-37: GICP_EVCNTRn bit assignments

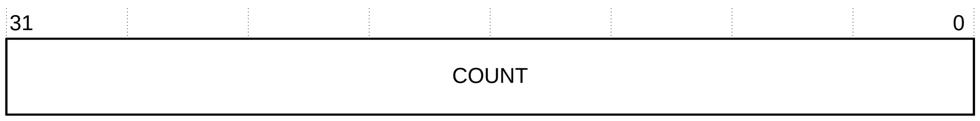


Table 5-60: GICP_EVCNTRn bit descriptions

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.

5.7.2 GICP_EVTYPERn, Event Type Configuration Registers

These registers configure which events that event counter `n` counts. The GIC-625 supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-38: GICP_EVTYPERn bit assignments

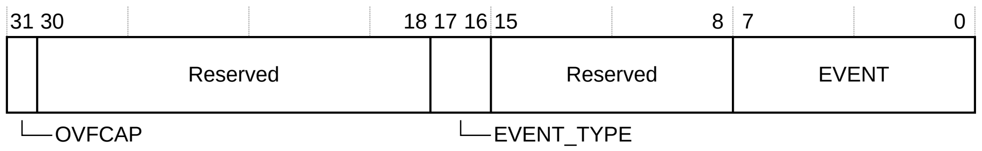


Table 5-61: GICP_EVTYPERn bit descriptions

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 **`0b11`** Reserved
[15:8]	-	Reserved
[7:0]	EVENT	Event identifier. See Table 5-62: GICP_EVTYPERn.EVENT field encoding on page 109. All events reset to an unknown value. Registers corresponding to unimplemented counters are RES0.

The following table shows the events that the GIC can count.

Table 5-62: GICP_EVTYPERn.EVENT field encoding

Event	Description	EventID	Filter
CLK	Clock cycle	`0x0`	None
CLK_NG	Clock cycle that prevents Q-Channel clock gating	`0x1`	None
-	Reserved	`0x2-0x3`	-

Event	Description	EventID	Filter
DN_MSG_PHY	Downstream message to core excluding PPIs	`0x4`	Target[10]
DN_SET_PHY	Set to core SPIs	`0x5`	Target/ID range[10]
DN_SET1OFN_PHY	Set to core, which is a 1 of N interrupt	`0x6`	Target/ID range[10]
-	Reserved	`0x7`	-
UP_ACT_SPI	Upstream activate for 1 of N SPIs only	`0x9`	Target/ID range[10]
UP_DEACT	Upstream deactivate. 1 of N SPIs only.	`0xD`	Target/ID range[10]
SGI_BRD	Broadcast SGI messages. Target = source.	`0x10`	Target/ID range
SGI_TAR	Targeted SGI messages. Target = source.	`0x11`	Target/ID range
SGI_ALL	All SGI messages. Target = source.	`0x12`	Target/ID range
SGI_ACC	Accepted SGI. Target = source.	`0x13`	Target/ID range
SPI_ENABLED	SPI enabled (new SPI or register access if pending)	`0x51`	ID range
SPI_DISABLED	SPI disabled (new SPI that is disabled or register access if pending)	`0x52`	ID range
ACC	Counter(`n` − 1) − counter(`n` − 2) every cycle. Prevents clock gating and Q-Channel clock gating.	`0x80`	None
OFLOW	Overflow of counter `n` − 1. Overflow counters cannot count overflows of the counters that are using the OFLOW event.	`0x81`	None
RLT_SPI_SET	Set to real-time SPIs	`0x90`	Target/ID[11] [12]
RLT_SPI_ACT	Upstream activate for real-time SPIs	`0x91`	Target/ID[11] [12]
RLT_SPI_REL	Upstream release for real-time SPIs	`0x92`	Target/ID[11] [12]
RLT_SPI_DEACT	Upstream deactivate for real-time SPIs	`0x93`	Target/ID[11] [12]
RLT_SPI_PEND_CNT	Pending real-time SPIs count at every cycle	`0x94`	Target/ID range[12]
RLT_SPI_ACT_CNT	Activate count for real-time SPIs at every cycle	`0x95`	Target/ID range[12]

[10] The GIC does not report this event when GICD_TYPER.No1N == 1.
[11] To filter this event, you must precisely match the target or ID, otherwise the behavior is unpredictable.
[12] This event always uses the filter in GICP_FR3, irrespective of the counter that is used. For example, if using counter 4, then the GIC uses GICP_FR3 and ignores GICP_FR4.

5.7.3 GICP_SVRn, Shadow Value Registers

These registers contain the shadow value of event counter `n`. The GIC-625 supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-39: GICP_SVRn bit assignments

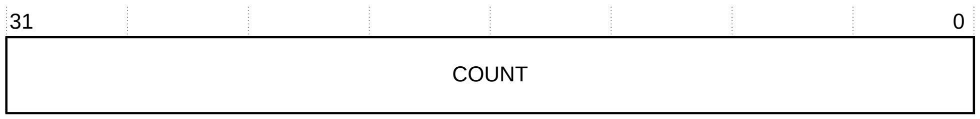


Table 5-63: GICP_SVRn bit descriptions

Bits	Name	Description
[31:0]	COUNT	Captured counter value. This field holds the captured counter values of the corresponding entry in GICP_EVCNTRn.

5.7.4 GICP_FRn, Filter Registers

These registers configure the filtering of event counter `n`. The GIC-625 supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-40: GICP_FRn bit assignments

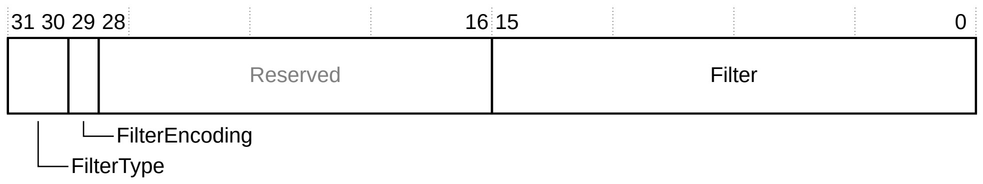


Table 5-64: GICP_FRn bit descriptions

Bits	Name	Description
[31:30]	FilterType	Filter type: • `0b00` = Filter on core • `0b01` = Filter on INTID • `0b10` = Reserved, no effect • `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 only associated with an exact match of the Filter-Type. 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

5.7.5 GICP_CNTENSET0, Counter Enable Set Register 0

These registers contain the counter enables for each event counter. The GIC-625 supports five event counters.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-41: GICP_CNTENSET0 bit assignments

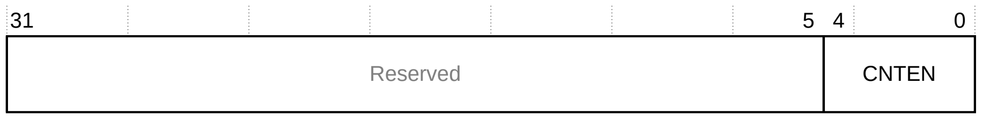


Table 5-65: GICP_CNTENSET0 bit descriptions

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.

5.7.6 GICP_CNTENCLR0, Counter Enable Clear Register 0

This register contains the counter disables for each event counter. The GIC-625 supports five event counters.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-42: GICP_CNTENCLR0 bit assignments

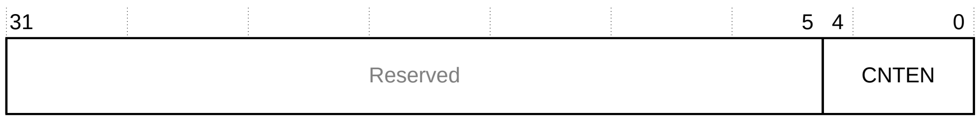


Table 5-66: GICP_CNTENCLR0 bit descriptions

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.

5.7.7 GICP_INTENSET0, Interrupt Contribution Enable Set Register 0

This register contains the set mechanism for the counter interrupt contribution enables. The GIC-625 supports five counters, n = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-43: GICP_INTENSET0 bit assignments

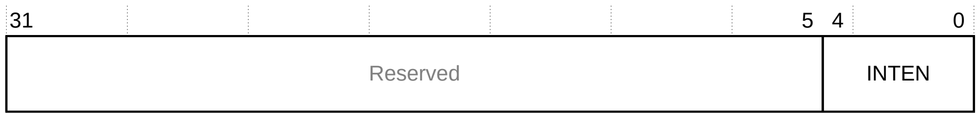


Table 5-67: GICP_INTENSET0 bit descriptions

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.

5.7.8 GICP_INTENCLR0, Interrupt Contribution Enable Clear Register 0

This register contains the clear mechanism for the counter interrupt contribution enables. The GIC-625 supports five counters, n = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-44: GICP_INTENCLR0 bit assignments

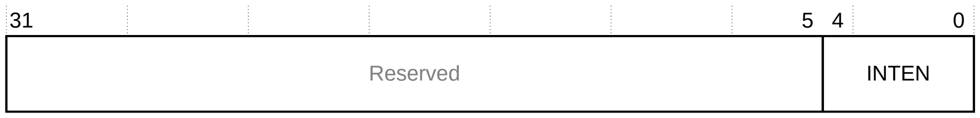


Table 5-68: GICP_INTENCLR0 bit descriptions

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.

5.7.9 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-625 supports five counters, n = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-45: GICP_OVSCLR0 bit assignments

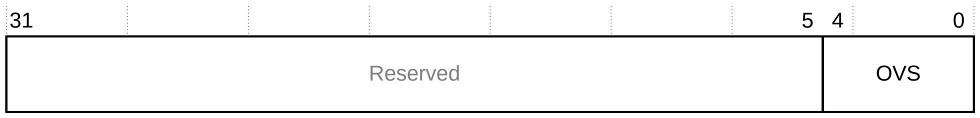


Table 5-69: GICP_OVSCLR0 bit descriptions

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 5.7.2 GICP_EVTYPERn, Event Type Configuration Registers on page 108.

5.7.10 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-625 supports five counters, `n` = 0-4.

Configurations

This register is available in all configurations.

Attributes

Width

32-bit

Functional group

See 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-46: GICP_OVSSET0 bit assignments

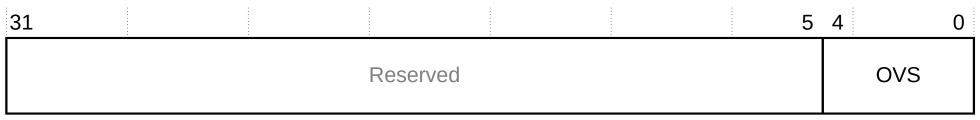


Table 5-70: GICP_OVSSET0 bit descriptions

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-625 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.

5.7.11 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 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-47: GICP_CAPR bit assignments

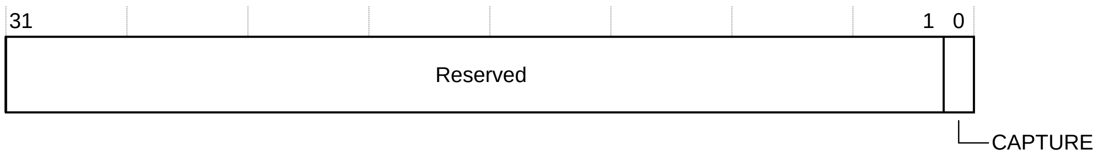


Table 5-71: GICP_CAPR bit descriptions

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 41 for information about other snapshot event triggers.	WO

5.7.12 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 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-48: GICP_CFGR bit assignments

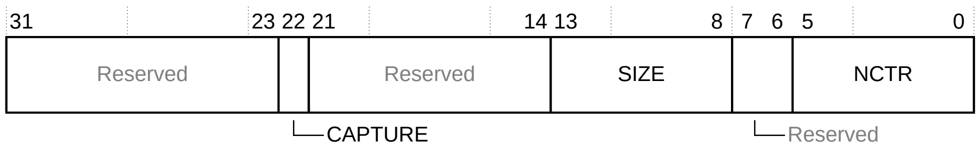


Table 5-72: GICP_CFGR bit descriptions

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.

5.7.13 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 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-49: GICP_CR bit assignments

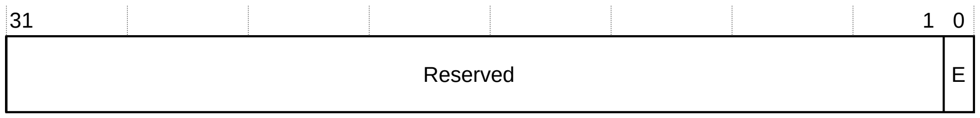


Table 5-73: GICP_CR bit descriptions

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.

5.7.14 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 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Table 5-74: GICP_IRQCR bit descriptions

Bits	Name	Description
[31:11]	-	Reserved, RAZ
[10:0]	SPIID	SPI ID. Returns 0 if an invalid entry is written. **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.

5.7.15 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 5.7 GICP register summary on page 107 for the address offset, type, and reset value of this register.

Usage constraints

There are no usage constraints.

Bit descriptions

Figure 5-50: GICP_PIDR2 bit assignments

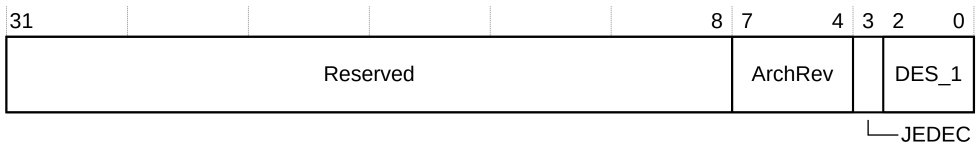


Table 5-75: GICP_PIDR2 bit descriptions

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.

Appendix A Getting started

There are some basic tasks that you must complete before you can start to use the GIC-625.

Each Redistributor must be powered on using its GICR_PWRR register to enable the Redistributors to be accessed, see 4.8.1 Redistributor power management on page 38 for more information.

When the GIC-625 is powered up, it must be programmed as the GICv3 and GICv4 Software Overview describes.

A.1 Other power management

The GIC-625 can be powered up and powered down using non-architectural protocols.

When powering down the GIC-625, software must preserve the state of the GIC-625, as defined in the Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4.

The implementation-defined powerdown sequence must:

1. Complete the powerdown sequence for all cores.
2. Set GICR_WAKER.Sleep to 1.
3. 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.

We recommend that you disable any interrupt sources before setting GICR_WAKER.Sleep.

When the GICR_WAKER.Quiescent bit is set, it is safe to power down the GIC-625. 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-625 if any interrupts arrive that must not be ignored.

For more information, see the GICv3 and GICv4 Software Overview.

A.2 Setting error recovery and fault handling options

Use the following procedures to set the error recovery and fault handling option.

Procedure

1. Write to GICT_ERR<c>MISC0.Count to preset the counter to any value.
   For example, to fire an interrupt on any correctable error, write `0xFF`, or to fire an interrupt on every second correctable error, write `0xFE`.
2. Assign a recorded uncorrectable ECC error to one of these options:
   - The fault-handling interrupt, **fault_int**, by setting GICT_ERR<n>CTLR.FI.
   - The error recovery interrupt, **err_int**, 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** are internally routed, the target interrupts must not have SPI 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** 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. Set route interrupts **fault_int** and **err_int** out as either:
   - Interrupt wires for situations where error recovery is handled by a core that does not receive interrupts directly from the GIC, such as a central system control processor.
   - Drive each interrupt internally by programming the associated GICT_ERRIRQCR<n> register. Each GICT_ERRIRQCR<n> register contains an ID field that must be programmed to 0 if internal routing is not required, or if internal routing is required, to a legally supported SPI ID.

     Note

     If the programmed ID value is less than 32, out of range, or not owned on chip for multichip configurations, the register updates to 0 and no internal delivery occurs.

A.3 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

This appendix describes the external input and output signals of the GIC-625.

B.1 Common control signals

The following table shows the GIC-625 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. Minimum of one cycle.
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-625 acknowledges that it is ready for MBIST testing.
[<domain>_]mbistreq	Input	MBIST mode request. Request to GIC-625 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[variable:0][13]	Input	Logical address. The width is based on the RAM with the largest number of words. You must drive the most significant bits to zero when accessing RAMs with fewer address bits.
[<domain>_]mbistindata[variable:0][13]	Input	Data in. Write data. Width that is based on the RAM with the largest number of data bits.
[<domain>_]mbistoutdata[variable:0][13]	Output	Data out. Read data. Width that is based on the RAM with the largest number of data bits.
[<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	

[13] The variable is configuration-dependent.

MBIST controller signals		
Signal	**Direction**	**Description**
[<domain>_]mbistarray[variable:0][13]	Input	Array selector. This signal controls which RAM array is accessed. 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-625 power control signals.

Signal definitions

Table B-2: Power control signals

Signal	Direction	Description
cpu_active[_<ppi_block>][_<bus>][<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 per core on the ICC bus. See 4.8.2 Processor core power management on page 39.
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** is sticky unless the Distributor is put into the gated state.

Distributor Q-Channel device interface for clock control		
Signal	**Direction**	**Description**
qreqn	Input	Q-Channel device interface for clock gating of the Distributor. **qreqn** is synchronized into the GIC-625. 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. **qreqn** is synchronized into the GIC-625. 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. **[<domain_>]clkqreqn** is synchronized into the GIC-625. This bus must be treated asynchronously.
[<domain_>]clkqacceptn	Output	
[<domain_>]clkqdeny	Output	
[<domain_>]clkqactive	Output	

B.3 Interrupt signals

The GIC-625 has interrupt signals for SPIs and PPIs.

Signal definitions

Table B-3: Interrupt signals

Signal	Direction	Description
rlt_spi[rlt_spi_wires−1:0] The *`rlt_spi_wires`* configuration parameter controls the number of real-time SPIs.	Input	This signal is the number of real-time SPI wires that the GIC supports. By default, SPIs are active-HIGH. The GIC provides top-level parameters so that an SPI can be active-LOW. To change an SPI to be active-LOW, set `RLT_SPI_INV<n> = 1`.
rlt_spi_r[rlt_spi_wires−1:0] The *`rlt_spi_wires`* configuration parameter controls the number of real-time SPIs.	Output	SPI output after synchronization and edge detection. Can be used for cross-domain pulse detection. If the *`RLT_SPI_R_INV`* parameter is set to 1, then it removes any inversion that *`RLT_SPI_INV[n]`* applies to individual SPIs.
ppi<n>[_<ppi_block>][_<bus>][<cpus>−1:0] If there are: • 16 PPIs per core, n is 16-31. • 32 PPIs per core, `n` is 16-31 and 1056-1071. • 48 PPIs per core, n is 16-31 and 1056-1087.	Input	PPI input wires for interrupt <n>. One bit per 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 top-level RTL parameters so that a PPI can be active-HIGH. The GIC also provides top-level RTL parameters so that a PPI can be synchronized to **clk**. 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_[_<ppi_block>][_<bus>]	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.

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, `<ppi_num>`, `<bus>`, `<cpuif_stream_width>`, and `<cpus>` are configuration options that are set using the *`ppi_ref`*, *`bus`*, *`cpuif_stream_width`*, and *`cpus`* parameters. See the *Arm® CoreLink™ GIC-625 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[_<ppi_num>][_<bus>]	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 **ICCTID** with values other than <*cpus*−1:0>, then the behavior is unpredictable.
icctvalid[_<ppi_num>][_<bus>]	Input	
icctdata[_<ppi_num>][_<bus>][<cpuif_stream_width>−1:0]	Input	
icctid[_<ppi_num>][_<bus>][<cpus>−1:0]	Input	
icctlast[_<ppi_num>][_<bus>]	Input	
icctwakeup[_<ppi_num>][_<bus>]	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[_<ppi_num>][_<bus>]	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[_<ppi_num>][_<bus>]	Output	
iritdata[_<ppi_num>][_<bus>][<cpuif_stream_width>−1:0]	Output	
iritdest [_<ppi_num>][_<bus>][<cpus>−1:0]	Output	
iritlast[_<ppi_num>][_<bus>]	Output	
iritwakeup[_<ppi_num>][_<bus>]	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 GIC-625 has an AMBA® ACE5-Lite subordinate interface. Software can use this interface to access the programming registers in the Distributor.

See the *Arm® CoreLink™ GIC-625 Generic Interrupt Controller Configuration and Integration Manual* for information about the signals that are present on the ACE5-Lite subordinate interface.

See 3.1.3.2 AMBA bus properties, GICD subordinate interface on page 21 for information about the ACE properties that the GIC supports.

B.6 Miscellaneous signals

The following table shows the GIC-625 miscellaneous signals.

Signal definitions

Table B-5: Miscellaneous signals

Signal	Direction	Description
fault_int	Output	Fault handling interrupt. The fault handling interrupt is defined in Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for Armv8-A. The GIC-625 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 4.10.3 Error recovery and fault handling interrupts on page 42.
err_int	Output	Error handling interrupt. The error handling interrupt is defined in Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for Armv8-A. The GIC-625 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 4.10.3 Error recovery and fault handling interrupts on page 42.
pmu_int	Output	PMU counter overflow interrupt. This signal is a level-sensitive interrupt. The GIC-625 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 41.
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 41 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.

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 22.

14 The variable is configuration-dependent.

Signal definitions

Table B-6: RAM I/O signals

Signal	Direction	Description
sgi_ram_in[SGI_RAM_IN_WIDTH−1:0]	Input	These I/O signals have no inherent functionality inside the GIC. The **tgt_spi_ram_*** signals are only present when GICD_TYPER.No1N==0.
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	

Appendix C Implementation-defined features

The GIC-625 implements features that are defined in the GICv3.1 architecture. Many of these features also have options in the GICv3.1 architecture, which determine behavior that is specific to the GIC-625. 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-625 uses. The table also gives references to sections within this manual that provide information about implementation-defined behavior that is specific to the GIC-625.

Table C-1: Declared implementation-defined features

GICv3.1 architecture feature	Architectural specification reference		Description
	Chapter	Section	
1 of N model	Introduction	Models for handling interrupts	See 4.7.4 SPI routing and 1 of N selection on page 36
INTIDs	Distribution and routing of interrupts	INTIDs	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 4.10 Reliability, Accessibility, and Serviceability on page 41.
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-625 supports 32 priority levels.
Effects of disabling interrupts	Physical interrupt handling and prioritization	Effect of disabling interrupts	Interrupts are set pending irrespective of the GICD_CTLR.EnableGrp* settings.

GICv3.1 architecture feature	Architectural specification reference		Description
	Chapter	Section	
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.

Appendix D Revisions

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

Table D-1: Issue 0000-01

Change	Location
First release	-

Table D-2: Differences between issue 0000-01 and issue 0000-02

Change	Location
Replaced the non-inclusive language for: • the type of ACE-Lite interface. The document now uses manager and subordinate interfaces. • the type of AXI5-Stream interface. The document now uses transmitter and receiver interfaces. • the type of GIC Stream interface. The document now uses requester and completer interfaces.	Throughout the document
Corrected the parameter names	3.1.1 Real-time SPI signals on page 18
Corrected the interface width options	3.2.1 GCI AXI5-Stream interface on page 24
Corrected the list of registers that require programming	• 4.6.1 PPI signals on page 34 • B.3 Interrupt signals on page 128
Corrected the corrupted SGI number formula	Table 4-5: SGI RAM errors, records 3-4 on page 45
Updated the GICT_ERR<n>MISC0.Data information	Table 4-6: TGT-SPI RAM errors, records 5-6 on page 46
Corrected the width and the reset value of GICD_ERRINSRn	5.2 Distributor registers (GICD/GICDA) summary on page 50
Updated the ERRINS1LOC and ERRINS2LOC descriptions	Table 5-14: GICD_ERRINSRn bit assignments for writes on page 64
Corrected the GICR_ICFGR1 reset value	5.5 Redistributor registers for SGIs and PPIs summary on page 84
Corrected the Count field description	Table 5-53: GICT_ERR<n>MISC0 bit descriptions on page 102
Corrected the GICP_PMDEVARCH reset value and the GICP_CIDR1 reset value	5.7 GICP register summary on page 107

Table D-3: Differences between issue 0000-02 and issue 0001-03

Change	Location
Added a configuration option that can remove support for 1 of N SPIs	3.1.5 Distributor configuration on page 22
Updated the No1N bit description	5.2.2 GICD_TYPER, Interrupt Controller Type Register on page 54
Added p1 to the Revision field description	• 5.2.3 GICD_IIDR, Distributor Implementer Identification Register on page 55 • 5.3.2 GICM_IIDR, Message-based Distributor Implementer Identification Register on page 72 • 5.4.2 GICR_IIDR, Redistributor Implementation Identification Register on page 75
Corrected the Version description for r0p0 and added r0p1 to the Version field description	5.5.10 GICR_CFGID1, Configuration ID1 Register on page 94

Change	Location
Updated the ACC and OFLOW descriptions	Table 5-62: GICP_EVTYPERn.EVENT field encoding on page 109
Added the Q-Channel signals for the Distributor and *GIC Cluster Interface* (GCI)	B.2 Power control signals on page 127