GIC-600AE (PL692)

Software Developer Errata Notice

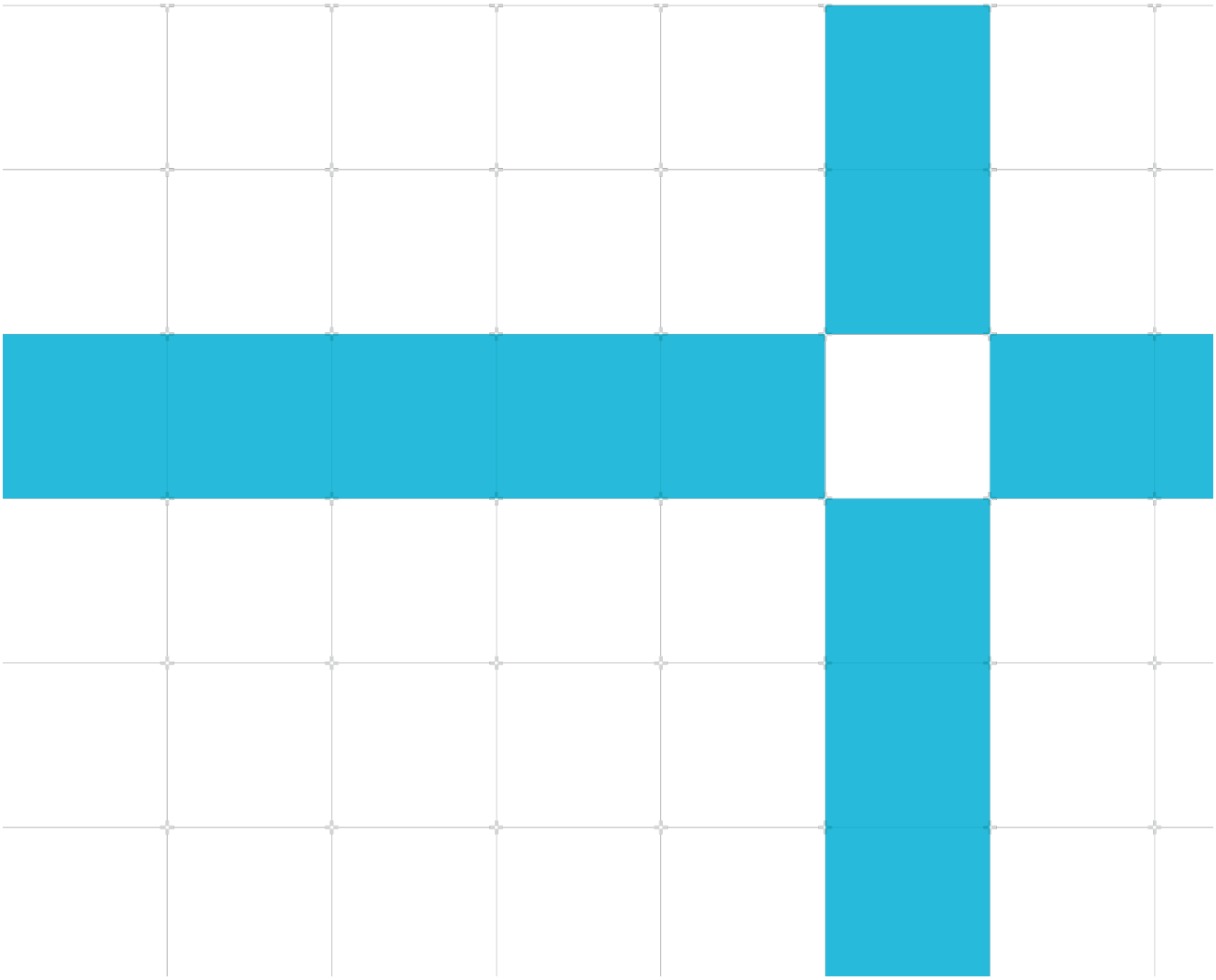
Date of issue: 23-Sep-2022

Non-Confidential

Document version: v6.0

Document ID: SDEN-1319947

This document contains all known errata since the r0p0 release of the product.



Non-confidential proprietary notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on GIC-600AE (PL692), 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.

If you find offensive language in this document, please email **terms@arm.com**.

Contents

Introduction

Scope

This document describes errata categorized by level of severity. Each description includes:

- The current status of the erratum.
- Where the implementation deviates from the specification and the conditions required for erroneous behavior to occur.
- The implications of the erratum with respect to typical applications.
- The application and limitations of a workaround where possible.

Categorization of errata

Errata are split into three levels of severity and further qualified as common or rare:

Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A (Rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B (Rare)	A significant error or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

Change Control

Errata are listed in this section if they are new to the document, or marked as "updated" if there has been any change to the erratum text. Fixed errata are not shown as updated unless the erratum text has changed. The **errata summary table** identifies errata that have been fixed in each product revision.

23-Sep-2022: Changes in document version v6.0

ID	Status	Area	Category	Summary
1568841	Updated	Programmer	Category B	SPI recall failure without subsequent trigger
2079287	Updated	Programmer	Category B	X-propagation can cause loss of lockstep out of reset
2420112	Updated	Programmer	Category B	Failure to forward highest priority interrupt
2439861	Updated	Programmer	Category B	FMU ping counter issues when used with low-power interface
2479150	Updated	Programmer	Category B	Clearing error with coincident firing can cause failure to report by safety mechanism
1685313	Updated	Programmer	Category C	SPI pipe does not always write back SEC corrections - including during SCRUB.
2023458	Updated	Programmer	Category C	Affinity3 field corrupted by cross-chip 32-bit writes to upper half of GICD_IROUTERn

05-Aug-2022: Changes in document version v5.0

ID	Status	Area	Category	Summary
2420112	New	Programmer	Category B	Failure to forward highest priority interrupt
2439861	New	Programmer	Category B	FMU ping counter issues when used with low-power interface
2479150	New	Programmer	Category B	Clearing error with coincident firing can cause failure to report by safety mechanism

12-Feb-2021: Changes in document version v4.0

ID	Status	Area	Category	Summary
2079287	New	Programmer	Category B	X-propagation can cause loss of lockstep out of reset
2023458	New	Programmer	Category C	Affinity3 field corrupted by cross-chip 32-bit writes to upper half of GICD_IROUTERn
2023462	New	Programmer	Category C	Target range check for MAPC/MOVALL ignores bits[51:48] of RDbase field

23-Sep-2020: Changes in document version v3.0

ID	Status	Area	Category	Summary
1685313	New	Programmer	Category C	SPI pipe does not always write back SEC corrections - including during SCRUB.
1713694	New	Programmer	Category C	Incorrect ERRSTATUS.SERR value
1733459	New	Programmer	Category C	64 bit writes to FMU registers may not come into effect

10-Sep-2019: Changes in document version v2.0

ID	Status	Area	Category	Summary
1373113	New	Programmer	Category B	False lockstep error reported due to qactive in lockstep comparator
1568841	New	Programmer	Category B	SPI recall failure without subsequent trigger
1339966	New	Programmer	Category C	Targeted Cross-Chip SGIs incorrectly generate a SGI_OOR error on the source chip
1445183	New	Programmer	Category C	fault on gicd_smen_fdc will report error even when SM is disabled
1452744	New	Programmer	Category C	DCHIPR reads 0 from chips that are not the Default Owner

16-Nov-2018: Changes in document version v1.0

No errata in this document version.

Errata summary table

The errata associated with this product affect the product versions described in the following table.

ID	Area	Category	Summary	Found in versions	Fixed in version
2479150	Programmer	Category B	Clearing error with coincident firing can cause failure to report by safety mechanism	r0p0, r0p1, r0p2	r0p3
2420112	Programmer	Category B	Failure to forward highest priority interrupt	r0p0, r0p1, r0p2	r0p3
2439861	Programmer	Category B	FMU ping counter issues when used with low-power interface	r0p0, r0p1, r0p2	r0p3
1568841	Programmer	Category B	SPI recall failure without subsequent trigger	r0p0, r0p1, r0p2	r0p3
1373113	Programmer	Category B	False lockstep error reported due to qactive in lockstep comparator	r0p0	r0p1
2079287	Programmer	Category B	X-propagation can cause loss of lockstep out of reset	r0p0, r0p1, r0p2	r0p3
1445183	Programmer	Category C	fault on gicd_smen_fdc will report error even when SM is disabled	r0p0	r0p1
1339966	Programmer	Category C	Targeted Cross-Chip SGIs incorrectly generate a SGI_OOR error on the source chip	r0p0	r0p1
2023462	Programmer	Category C	Target range check for MAPC/MOVALL ignores bits[51:48] of RDbase field	r0p0, r0p1, r0p2, r0p3	Open
2023458	Programmer	Category C	Affinity3 field corrupted by cross-chip 32-bit writes to upper half of GICD_IROUTERn	r0p0, r0p1, r0p2	r0p3
1685313	Programmer	Category C	SPI pipe does not always write back SEC corrections - including during SCRUB.	r0p0, r0p1, r0p2	r0p3
1452744	Programmer	Category C	DCHIPR reads 0 from chips that are not the Default Owner	r0p0	r0p1
1733459	Programmer	Category C	64 bit writes to FMU registers may not come into effect	r0p0, r0p1	r0p2
1713694	Programmer	Category C	Incorrect ERRSTATUS.SERR value	r0p0, r0p1	r0p2

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

2479150
Clearing error with coincident firing can cause failure to report by safety mechanism

Status

Fault Type: Programmer CAT B
Fault Status: Present in r0p0, r0p1, r0p2. Fixed in r0p3.

Description

When the GICD acknowledges the reporting of a fault by a remote GIC block (ITS, PPI, Wake, Col), the remote block's safety mechanism (SM) could get into a state in which subsequent errors from the affected SM are not reported.

Conditions

For the issue to occur, the fault's ack/clear from the GICD (initiated by software) must arrive at the remote block's safety mechanism (SM) on the same cycle in which a subsequent new error is reported.

Configurations

All GIC-600AE FuSa configurations with remote blocks (PPI, ITS, Wake, Col). These blocks contain SMs that detect errors and report them to the GICD. Errors detected and reported by GICD SMs are not affected.

Implications

If the condition occurs, subsequent errors might not be reported over the AXI-Stream interface until either:

- a clear is received (without an error being flagged by the SM)
- the SM is disabled and then re-enabled.

Workaround

Reset the GIC when an error is detected. For this to work reliably, CEs should be treated as UEs by ensuring that the FMU_ERR<n>CTLR.CE_EN field is cleared.

2420112
Failure to forward highest priority interrupt

Status

Fault Type: Programmer CAT B
Fault Status: Present in r0p0, r0p1, r0p2. Fixed in r0p3.

Description

The Distributor (GICD) has a buffer where it stores the next SET or CLEAR packet to be sent to any CPU.
When the GIC is waiting to send a message to a CPU, if it re-evaluates that CPU and finds a new higher priority message, then it fails to update the request and assumes the new packet has been sent.

There are 2 possible failure modes:

Part 1 : If the packet to be sent is a SET packet then a higher priority SET may not be sent until an unblocking event occurs.

Part 2 : If the packet is a CLEAR (caused by an SPI recall) and the same interrupt is released from the CPU, then a further SET packet may be delayed until an unblocking event occurs.
Note: Relevant releases can only be generated if the corresponding cpu_group_enable is being disabled in the CPU.

Unblocking events are any of the following:

- Toggling any GICR_CTLR.DPG<x> bit of the impacted CPU
- Toggling any cpu_group_enable bit of the impacted CPU
- Toggling any gicd_group_enable
- Activate/Release of the outstanding interrupt on the impacted CPU
- Another interrupt arrives that is accepted as one of the top 5 (3 if no LPI) interrupts for the impacted CPU.

Configurations impacted

All.

Conditions

Interrupt in the following description refers to an LPI (if configured) or an SPI.

Part 1: (SET)

1. A CPU has no SPI or LPI sent to it.
   Note this can occur after the activation of an interrupt. It does not mean there are no interrupts

pending for a CPU in the GIC.
2. The GIC identifies a new interrupt (A) and generates a SET packet.
3. A new higher priority interrupt (B) arrives, so the GIC attempts to re-evaluate the CPU but fails to send the new interrupt because the previous SET packet has not yet been sent.
4. Interrupt (B) is not sent until an unblocking event occurs.

Part 2: (CLEAR)

1. A CPU has an SPI sent to it and no other pending interrupts identified.
2. The SPI is recalled because it becomes disabled or non-pending, causing a CLEAR packet to be created.
3. The corresponding CPU group enable is cleared, causing SPI to be released.
4. The GIC identifies a new interrupt (C) for the same CPU and attempts to send it, but fails as the CLEAR packet has not yet been issued.
5. Interrupt (C) is not sent until an unblocking event occurs.

Implications

Part 1: (SET)

If the Priority Mask Register (PMR) is not being used, then this appears as a temporary miss-prioritization with no real impact.
However, if the PMR is being used to the first SET while waiting for the second, the system may hang as the second SET will not necessarily be delivered in a finite time.

In cases where an OS does use the PMR, it is expected that the value will ultimately be relaxed to allow all interrupts to be serviced, which would allow servicing of any stalled interrupts to continue. This is the case in upstream Linux, which only uses PMR to create pseudo-NMIs for profiling purposes.

Part 2: (CLEAR)

If the interrupt being targeted by the CLEAR is released from the CPU before the CLEAR is sent to the CPU, then a subsequent SET packet may not be delivered in a finite time.

Workaround

Part 1: (SET)

If not using the PMR, then no workaround is expected to be required.

However if PMR functionality must be used, then either:

- Periodically toggle GICR_CTLR.DPG<x> to ensure that interrupts are delivered. The required frequency will be a function of system interrupt latency tolerance.
- If not using LPIs, then use GICD_I(S|C)ENABLERn to model PMR functionality by disabling interrupts that would otherwise be masked. Note there is no need to poll GICD_CTLR.RWP in this case.

Part 2: (CLEAR)

Software should issue a **DSB** and toggle GICR_CTLR.DPG<x> after clearing the corresponding cpu_group_enable.

2439861
FMU ping counter issues when used with low-power interface

Status

Fault Type: Programmer CAT B
Fault Status: Present in r0p0, r0p1, r0p2. Fixed in r0p3.

Description

The GIC has a ping function which periodically pings remote block to ensure that the communications channels are still functional.

If the GIC is requested to enter low-power mode (QSTOPPED) while the GIC is about to issue a ping message (but is otherwise idle), then it incorrectly enters QSTOPPED.

Conditions

The erratum can occur if the following conditions are met at the same time:

1. The GIC is programmed to issue pings and it starts to issue a ping to any remote block.
2. The GIC is idle, apart from the ping logic.
3. A low-power entry request is received on the Q-Channel.

Configurations impacted

Any.

Implications

There are two implications:

- The ping interface will stop and not issue or report further pings until it is reprogrammed.
- The GIC may enter QSTOPPED with a ping message on an AXI-Stream bus. This may remain on the bus with the **VALID** signal HIGH and appear as multiple messages being issued on the bus.

If this occurs, the **qactive** signal will be asserted.

Workaround

If using the Q-Channel, then software should program GICD_CTLR.QDENY during normal operation.

To enter low-power mode, software should:

1. Disable the ping functionality
2. Clear GICD_CTLR.QDENY
3. Issue low power entry request.

1568841
SPI recall failure without subsequent trigger

Status:

Fault Type: Programmer CAT B
Fault Status: Present in r0p0, r0p1, r0p2. Fixed in r0p3.

Description

If an interrupt is identified as one of the top priority enabled interrupts for a CPU and is sent to the target cache, then there is a single-cycle window where if the interrupt is reprogrammed then the recall requirement is logged but not performed until the next external trigger.

The triggers can be any of:

- Activation, release, or deactivation of any SPI.
- A state change on any SPI signal.
- Register programming or a CPU group enable change.

Implications

There are two implications:

- If software changes a GICD_IROUTER register, then the ACE-Lite slave interface might not respond until the next trigger occurs or the CPU services the interrupt.
- If software programs registers other than ICENABLER, then the duration when the GIC uses old programming might extend until after the next trigger. The GIC does not use old programing more than once, or go back to old programming once it uses the new programming.

Workaround

Disable SPIs by writing to ICENABLER<n> before reprogramming them, especially if rerouting them by programming GICD_IROUTER.
This workaround ensures that all reprogramming of an SPI is atomic, and is the standard behavior of the Linux driver.

1373113
False lockstep error reported due to qactive in lockstep comparator

Status

Fault Type: Programmer CAT B
Fault Status: Present in r0p0, Fixed in r0p1

Description:

A false lockstep comparison error is generated by GIC-600AE. The ACE Switch lockstep comparison incorrectly includes the qactive signal.

The lockstep comparator expects that the qactive and qactive_fdc signals are two-cycle delayed with respect to each other. In point-to-point mode, these signals are not two-cycle delayed, and the lockstep comparator reports a lockstep error.

Correct behavior:

The lockstep error is not reported by the GIC-600AE.

Conditions:

This is configuration dependent and will show up whenever the icdiwakeup signal is asserted, because icdiwakeup feeds into qactive.

Configurations affected:

The false lockstep error is reported when all of the following configuration parameters are set:

- bypass_ports = 1, which instantiates the ITS ACE bypass switch.
- fusa_axis_int_busprot_type= 0, which selects point-to-point mode for AXI4-Stream protection.

Implication:

A false lockstep error is flagged by the Fault Management Unit inside the GIC-600AE through the ERI or FHI interrupt.

Workaround:

Software workaround:

Disable the Safety mechanism #8 (ACE bypass switch error) in the ITS by writing to FMU_SMEN the value "32'h0800_{ITS#}00", where ITS# is the ITS ID in 8 bits. Arm expects the impact on the SPFM to be low, and that the majority of systems will still meet their functional safety requirements with this safety mechanism disabled.

Hardware workaround:

Configure fusa_axis_int_busprot_type = 1, which uses CRC protection on interface.

2079287
X-propagation can cause loss of lockstep out of reset

Status:

Fault Type: Programmer CAT B
Fault Status: Present in r0p0, r0p1, r0p2. Fixed in r0p3.

Description

X-propagation can cause loss of lockstep in GIC-600AE.

Conditions

Multiple interrupts become pending and enabled together.

Configurations impacted

All.

Implications

Lockstep is lost between the primary and secondary GIC instances, resulting in a fault being reported.

Workaround

Write **`0xFFFFFFFF`** to the GICD_ICENABLER1 register, before initializing the GIC.

Category B (rare)

There are no errata in this category.

Category C

1445183
fault on gicd_smen_fdc will report error even when SM is disabled

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, Fixed in r0p1

Description:

The gicd_smen is a signal inside GIC from the Fault management unit (FMU) going to the Safety mechanism which can be programmed by the user to tell the safety mechanism to not report faults when detected. If there is a fault on this signal (gicd_smen), then a lockstep error is still reported by the GIC

Correct behaviour

The FMU should not have reported the fault

Conditions

For this errrata to have an effect all the following must be happen

1) Fault on signal gicd_smen . This is rare as it would be 1 signal out of all the signals inside GIC (order of 1/million).

2) User should have disabled that safety mechanism

Configurations affected:

All

Implication:

A lockstep error would flagged by the fault management unit inside the GIC through the ERI or FHI interrupt

Workaround:

No workaround is required

1339966
Targeted Cross-Chip SGIs incorrectly generate a SGI_OOR error on the source chip

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, Fixed in r0p1

Description

An targeted (non-broadcast) SGI targeted at a remote chip will generate an SGI OOR error which will be reported in T&D Record 0 of the source chip.

The SGIs is still delivered correctly.

Implication

Error Record 0 will be flooded with false SGI errors. This may mask other real software errors that are generating as only a single error is recorded unless the record is cleared.

If record 0 is programmed to generate a fault_handling or error_recovery interrupt then spurious error interrupts will be generated.

There is no impact on tracking ECC or ITS software errors and no impact on the architectural operation of the GIC.

Workaround

Any of the following workarounds can be applied:

a) Do not enable the interrupts generated from error record 0 and ignore error record 0. (This is the reset behaviour of the GIC)

b) If the interrupts are enabled clear error record 0 after every remote SGI (based on receiving the error interrupt)

c) If interrupts are enabled (or software plans to periodically check the error record status registers) during software debug then trap writes to the SGI generation registers and disable before generating each Cross-Chip targeted SGI.

2023462
Target range check for MAPC/MOVALL ignores bits[51:48] of RDbase field

Status:

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, r0p1, r0p2, r0p3. Open

Description

The ITS ignores bits[51:48] of the RDbase (target) field of **`MAPC`**/**`MOVALL`** commands.

ITS programing should only be available to privileged software.

Conditions

Target field [51:48] of the RDbase field are nonzero during a **`MAPC`** or **`MOVALL`**

Implications

A command issued with an out-of-range target might not be detected, and it might execute with a truncated TGT field.

Workarounds

Software should only use legal targets that only use the lower bits of the RDbase field.

2023458
Affinity3 field corrupted by cross-chip 32-bit writes to upper half of GICD_IROUTERn

Status:

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, r0p1, r0p2. Fixed in r0p3.

Description

The upper 32 bits of the GICD_IROUTERn registers contain the Affinity3 field, and the whole register should be accessible from the GICD address space of any connected chip using 32-bit or 64-bit accesses.

However, using a 32-bit write to the upper half of a GICD_IROUTERn for an SPI that is owned on a different chip, results in the Affinity3 field being set to 0 rather than the programmed value.

Access to GIC registers should be limited to privileged software only.

Configurations impacted

GIC configurations that include multiple chips and an affinity3 width of greater than 0, that is, when:

- The **`chip_count`** configuration parameter is set to 1, and
- The **`chip_affinity_select_level`** configuration parameter is set to 3.

Note that AArch32 systems do not support nonzero Affinity3 values, so this erratum only impacts AArch64 systems.

To trigger this erratum, software must have already completed multiple 64-bit accesses, to have successfully connected the chips.

Conditions

Software issues a 32-bit write to the upper half of a GICD_IROUTERn for an SPI that is owned on another chip.

Implications

Impacted SPIs are not routed to the expected CPUs. Depending on the system topology and programming, the SPI might be delivered to a CPU on chip 0 with matching Affinity2, Affinity1 and Affinity0, or the SPI might not be delivered.

Workaround

There are 2 possible workarounds:

- Always use 64-bit writes when setting the Affinity3 field in GICD_IROUTERn. 32-bit writes are acceptable when setting GICD_IROUTERn.IRM. This workaround is the expected behavior.
- Program GICD_IROUTERn registers through the GICD or GICDA registers space, on the chip where the SPI is currently owned according to the GICD_CHIPR registers.

1685313
SPI pipe does not always write back SEC corrections - including during SCRUB.

Status:

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, r0p1, r0p2. Fixed in r0p3.

Description

Single-bit errors that are detected in the SPI pipe are always detected and reported but are not always corrected immediately after detection.

General searching and RAM scrubs do not cause RAM writebacks for single-bit errors.

Double-bit error detection is not affected.

Implications

There is no impact on the architectural operation of the GIC but the MTBF might increase.

A RAM scrub (setting GICD_FCTLR.SIP to 1) only reports errors and does not correct single-bit errors.

Workaround

If a scrub reports a *Single-bit Error Correct* (SEC) error or it reports multiple SEC errors on the same SPI line during operation, then to clear the fault, write to any register which impacts that SPI such as GICD_IGROUPRn. This workaround is successful unless it is genuine stuck-at-fault.

The register write can occur in the Secure or Non-secure world, because the GIC performs the error correction in the RAM even if the security check fails.

1452744
DCHIPR reads 0 from chips that are not the default owner

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, Fixed in r0p1

Description

The DCHIPR register contains 2 fields

1) RTOwner - Which chip is the current owner of the Routing table (also know as Default Owner or Crown Socket)

2) PUP - Indicates that the RT owner is currently performing an update.

The only reason for writing this register is to change the RTOwner if planning to power down the RTOwner Socket.

Operations to move the owner will complete successfully, however PUP may incorrectly indicate that the update has completed whilst still in progress

Implications

Software may think PUP has dropped early.

If software attempts to start a new access e.g. chip offline before PUP has really dropped then the new access will be rejected.

Rejected writes to update chip state will be detectable as the relevant CHIPR register will not record the updated value.

Workaround

If changing the RT Owner is really required, then software should poll DCHIPR on both the new and old sockets to ensure that PUP is recoded as 0 on both, before attempting further Routing table operations.

1733459

64 bit writes to FMU registers may not come into effect

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, r0p1, Fixed in r0p2

Description

The Fault Management Unit (FMU) registers are protected using a lock and key mechanism. Once unlocked, the registers are locked again after a secure write to any register. Some FMU register are 64 bit registers wide, however the APB bus interface is 32 bits wide. The upper 32 bits of these registers is RESERVED or have read-only fields.

The design currently expects the software to perform 32 bit access to these 64 bit registers so that a KEY write can unlock the register before it updates that register.

If software makes a 64bit write to a register with data[63:0], the access will be split by the APB interconnect into two APB write of 32 bits each and will be forwarded to the FMU APB interface.

If these 32 bit writes reach the FMU interface in order "data[31:0]" followed by "data[63:32] " then lower 32 bits of the register (which contains meaningful contents) gets updated correctly. The upper 32 bits of the register would not be updated as the KEY would lock . This is not an issue as the upper 32 bits is RESERVED or read-only.

However, if the writes reaches FMU interface in order "data[63:32]" followed by "data[31:0]", then the lower 32 bits of the register would not be updated.

Configurations affected

All configurations

Conditions

All the following must be true

1. Software must perform 64 bit write
2. The APB interconnect must split the 64 bit write into two 32 bit access with the order "data[63:32]" followed by "data[31:0]"

Implication

The programming of FMU may not be successful.

Note: The FMU registers are mapped into "device memory". Typically, the APB interconnect would not reorder the 64 bit access in this specific order. For e.g. ARM NIC400 APB interconnect always orders split writes as data[31:0] followed by data[63:32] in which case the system is not affected by errata.

Workaround

Hardware Workaround

The APB interconnect on receiving a 64 bit write with data[63:0] should split it into two APB writes in the order data[31:0] followed by data[63:32]

Software Workaround

The software should use 32 bit access to FMU registers.

1713694

Incorrect ERR<n>STATUS.SERR value

Status

Fault Type: Programmer CAT C
Fault Status: Present in r0p0, r0p1, Fixed in r0p2

Description

Fault Management Unit (FMU) reports random hardware faults detected by asserting an interrupt. On receipt of an interrupt, the error recovery handling routine inspects the error records of the FMU to take appropriate actions. As part of this inspection the software performs the following steps

1) Read the ERRGSR register to find which error record number is reporting an error

2) Read ERR<n>STATUS register to get more information

a) Bit[31] : V : would indicate this error record is in error

b) bit[25:24] : would indicate if this is a correctable error

c) bit[29] : would indicate if this is an uncorrectable error

d) bit[15:8] : identify the source of the error

e) bit[7:0]: SERR : provides additional information on architecturally defined primary code. When there is no error this field return 8'd0 When there is an error this field should return a non-zero value.

The defect is that this field always return 8'd0 even in case of error being reported.

Conditions

When error is reported, this field return incorrect value

Configurations affected

All configurations

Implications

Impact should be low as this is one of the fields which provides more additional information on the error. All the other information regarding the error is still valid (2.a to 2.d above). It may be confusing to software to read a error record indicating an error with this field set to 8'd0 .

Workaround

Software should ignore this field. Use all other information described in points 2.a to 2.d above.