



PL690 Generic Interrupt Controller

Software Developer Errata Notice

Date of issue: 18-May-2022

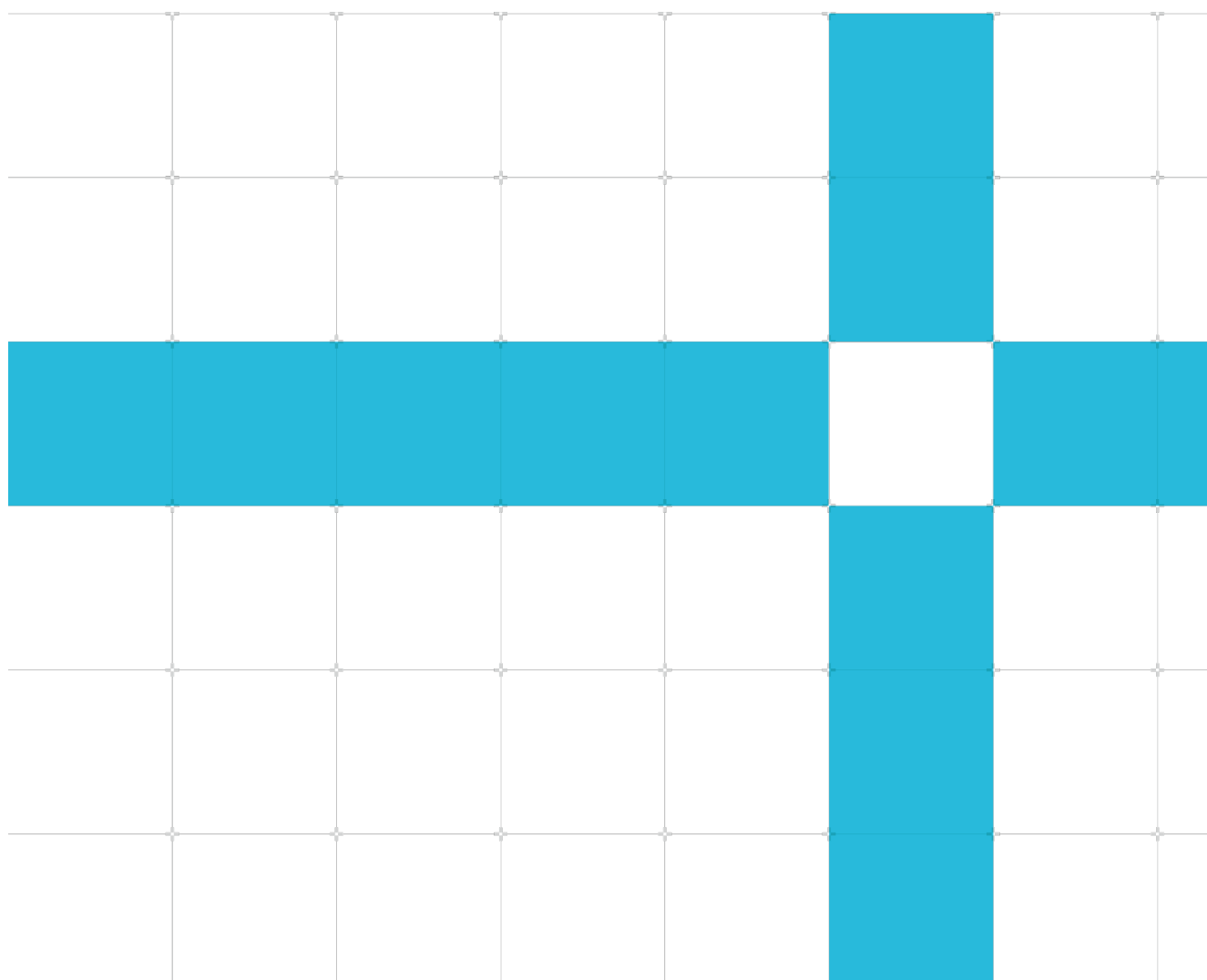
Non-Confidential

Copyright © 2018-2022 Arm® Limited (or its affiliates). All rights reserved.

Document version: v12.0

Document ID: SDEN-892601

This document contains all known errata since the r0p2 release of the product.



Non-confidential proprietary notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

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

This document may include technical inaccuracies or typographical errors.

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

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

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2018-2022 Arm® Limited (or its affiliates). All rights reserved.

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 PL690 Generic Interrupt Controller, 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	6
Scope	6
Categorization of errata	6
Change Control	7
Errata summary table	9
Errata descriptions	11
Category A	11
Category A (rare)	11
Category B	12
2384374 Failure to forward highest priority interrupt.	12
1717652 Wake_request may not be delivered if multiple cores are woken by PPIs at the same time	15
1494863 SPI recall failure without subsequent trigger	17
1196726 CLEAR command may continue after a SYNC	18
1118145 QACTIVE may not be driven HIGH on qreqn_its_<n> transition	19
1109146 GICD_CTLR.RWP may be stuck at 1 if already 1 when rt_owner changes	20
1014611 Lower or same priority PPI/SGI may not be sent until after deactivate of previous interrupt	21
996333 Disabled Secure SPIs blocking LPIs	22
990914 Issues around PT triggering affecting LPI delivery	23
988378 Target cache presenting incorrect priorities when DS is set for filter	24
957913 Interrupt on Target_Cache Return interface can be missed by ITS commands	25
907913 PT coarse map may be written back when invalid following sleep completion and wakeup in PT	27
Category B (rare)	28
975678 Broadcast GEN_SGI generates interrupt for the source CPU when >10 CPUs	28
931156 ITS not ignoring corrupted RDATA when ITS receives an ERROR response	29
Category C	30
2023459 Target range check for MAPC/MOVALL/VMAPP/VMOVP ignores bits[51:48] of RDbase field	30
2023457 Affinity3 field corrupted by cross-chip 32-bit writes to upper half of GICD_IROUTERn	31
1621321 SPI pipe does not always write back single-bit error corrections - including during scrub	33
1451068 DCHIPR reads 0 from chips that are not the default owner	34

1335020	Targeted Cross-Chip SGIs incorrectly generate a SGI_OOR error on the source chip	35
1109171	Possible SPI corruption if GICD_IERRR<n> written to 1 without being read first in multi-chip configs	36
1104459	ITS debug overflow not recorded in trace registers	37
1098957	UNPREDICTABLE behavior on failed attempt to connect chips	38
1071611	DPG might allow a single 1ofN interrupt to be sent after being set	39
1049857	PMU filtering for Deactivate Event incorrect	40
1003544	PPI does not mask out GRPMOD when GICD_CTLR.DS=1	41
963506	Deactivate - Activate ordering violation may leave interrupt not active	42
930020	IRQCR accesses trigger out of range SPI block software errors	43

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.

18-May-2022: Changes in document version v12.0

ID	Status	Area	Category	Summary
2384374	New	Programmer	Category B	Failure to forward highest priority interrupt.

20-Jan-2021: Changes in document version v11.0

ID	Status	Area	Category	Summary
2023457	New	Programmer	Category C	Affinity3 field corrupted by cross-chip 32-bit writes to upper half of GICD_IROUTERn
2023459	New	Programmer	Category C	Target range check for MAPC/MOVALL/VMAPP/VMOVP ignores bits[51:48] of RDbase field

27-May-2020: Changes in document version v10.0

ID	Status	Area	Category	Summary
1717652	New	Programmer	Category B	Wake_request may not be delivered if multiple cores are woken by PPIs at the same time
1621321	New	Programmer	Category C	SPI pipe does not always write back single-bit error corrections - including during scrub

02-Aug-2019: Changes in document version v9.0

ID	Status	Area	Category	Summary
1494863	New	Programmer	Category B	SPI recall failure without subsequent trigger

25-Apr-2019: Changes in document version v8.0

ID	Status	Area	Category	Summary
1451068	New	Programmer	Category C	DCHIPR reads 0 from chips that are not the default owner

08-Feb-2019: Changes in document version v7.0

No new or updated errata in this document version.

30-Nov-2018: Changes in document version v6.0

No new or updated errata in this document version.

25-Jul-2018: Changes in document version v5.0

ID	Status	Area	Category	Summary
1196726	New	Programmer	Category B	CLEAR command may continue after a SYNC

14-Jun-2018: Changes in document version v4.0

No new or updated errata in this document version.

23-Apr-2018: Changes in document version v3.0

ID	Status	Area	Category	Summary
1109146	New	Programmer	Category B	GICD_CTLR.RWP may be stuck at 1 if already 1 when rt_owner changes
1118145	New	Programmer	Category B	QACTIVE may not be driven HIGH on qreqn_its_ transition
1071611	New	Programmer	Category C	DPG might allow a single 1ofN interrupt to be sent after being set
1098957	New	Programmer	Category C	UNPREDICTABLE behavior on failed attempt to connect chips
1104459	New	Programmer	Category C	ITS debug overflow not recorded in trace registers
1109171	New	Programmer	Category C	Possible SPI corruption if GICD_IERRR written to 1 without being read first in multi-chip configs

12-Jan-2018: Changes in document version v2.0

No new or updated errata in this document version.

11-Jan-2018: Changes in document version v1.0

ID	Status	Area	Category	Summary
907913	New	Programmer	Category B	PT coarse map may be written back when invalid following sleep completion and wakeup in PT
957913	New	Programmer	Category B	Interrupt on Target_Cache Return interface can be missed by ITS commands
988378	New	Programmer	Category B	Target cache presenting incorrect priorities when DS is set for filter
990914	New	Programmer	Category B	Issues around PT triggering affecting LPI delivery
996333	New	Programmer	Category B	Disabled Secure SPIs blocking LPIs
1014611	New	Programmer	Category B	Lower or same priority PPI/SGL may not be sent until after deactivate of previous interrupt
931156	New	Programmer	Category B (rare)	ITS not ignoring corrupted RDATA when ITS receives an ERROR response
975678	New	Programmer	Category B (rare)	Broadcast GEN_SGI generates interrupt for the source CPU when >10 CPUs
930020	New	Programmer	Category C	IRQCR accesses trigger out of range SPI block software errors
963506	New	Programmer	Category C	Deactivate - Activate ordering violation may leave interrupt not active
1003544	New	Programmer	Category C	PPI does not mask out GRPMOD when GICD_CTLR.DS=1
1049857	New	Programmer	Category C	PMU filtering for Deactivate Event incorrect

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
2384374	Programmer	Category B	Failure to forward highest priority interrupt.	r0p2, r0p3, r1p1, r1p2, r1p3, r1p4, r1p5, r1p6	Open
1717652	Programmer	Category B	Wake_request may not be delivered if multiple cores are woken by PPIs at the same time	r0p2, r0p3, r1p2, r1p3, r1p4, r1p5, r1p6	Open
1494863	Programmer	Category B	SPI recall failure without subsequent trigger	r0p2, r0p3, r1p2, r1p3, r1p4, r1p5, r1p6	Open
1196726	Programmer	Category B	CLEAR command may continue after a SYNC	r0p2, r0p3, r1p2, r1p3	r1p4
1118145	Programmer	Category B	QACTIVE may not be driven HIGH on qreqn_its_ transition	r1p2	r1p3
1109146	Programmer	Category B	GICD_CTLR.RWP may be stuck at 1 if already 1 when rt_owner changes	r1p2	r1p3
1014611	Programmer	Category B	Lower or same priority PPI/SGL may not be sent until after deactivate of previous interrupt	r0p2, r0p3	r1p2
996333	Programmer	Category B	Disabled Secure SPIs blocking LPIs	r0p2	r0p3
990914	Programmer	Category B	Issues around PT triggering affecting LPI delivery	r0p2	r0p3
988378	Programmer	Category B	Target cache presenting incorrect priorities when DS is set for filter	r0p2	r0p3
957913	Programmer	Category B	Interrupt on Target_Cache Return interface can be missed by ITS commands	r0p2	r0p3
907913	Programmer	Category B	PT coarse map may be written back when invalid following sleep completion and wakeup in PT	r0p2	r0p3
975678	Programmer	Category B (rare)	Broadcast GEN_SGI generates interrupt for the source CPU when >10 CPUs	r0p2	r0p3
931156	Programmer	Category B (rare)	ITS not ignoring corrupted RDATA when ITS receives an ERROR response	r0p2	r0p3
2023459	Programmer	Category C	Target range check for MAPC/MOVAL/VMAPP/VMOVP ignores bits[51:48] of RDbase field	r0p2, r0p3, r1p1, r1p2, r1p3, r1p4, r1p5, r1p6	Open
2023457	Programmer	Category C	Affinity3 field corrupted by cross-chip 32-bit writes to upper half of GICD_IROUTERn	r1p1, r1p2, r1p3, r1p4, r1p5, r1p6	Open

ID	Area	Category	Summary	Found in versions	Fixed in version
1621321	Programmer	Category C	SPI pipe does not always write back single-bit error corrections - including during scrub	r0p2, r0p3, r1p2, r1p3, r1p4, r1p5, r1p6	Open
1451068	Programmer	Category C	DCHIPR reads 0 from chips that are not the default owner	r1p2, r1p3, r1p4, r1p6	Open
1335020	Programmer	Category C	Targeted Cross-Chip SGIs incorrectly generate a SGI_OOR error on the source chip	r1p2, r1p3, r1p4	r1p6
1109171	Programmer	Category C	Possible SPI corruption if GICD_IERRR written to 1 without being read first in multi-chip configs	r1p2	r1p3
1104459	Programmer	Category C	ITS debug overflow not recorded in trace registers	r0p2, r0p3, r1p2	r1p3
1098957	Programmer	Category C	UNPREDICTABLE behavior on failed attempt to connect chips	r1p2	r1p3
1071611	Programmer	Category C	DPG might allow a single 1ofN interrupt to be sent after being set	r0p2, r0p3	r1p2
1049857	Programmer	Category C	PMU filtering for Deactivate Event incorrect	r0p2, r0p3	r1p2
1003544	Programmer	Category C	PPI does not mask out GRPMOD when GICD_CTLR.DS=1	r0p2	r0p3
963506	Programmer	Category C	Deactivate - Activate ordering violation may leave interrupt not active	r0p2	r0p3
930020	Programmer	Category C	IRQCR accesses trigger out of range SPI block software errors	r0p2	r0p3

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

2384374

Failure to forward highest priority interrupt.

Description

The Distributor (GICD) has a buffer where it stores the next SET or CLEAR packet to be sent to any CPU.

If the GIC is waiting to send a message to a CPU, 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 when it should be 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:

1. Toggling any GICR_CTLR.DPG<x> bit of the impacted CPU
2. Toggling any cpu_group_enable bit of the impacted CPU
3. Toggling any gicd_group_enable
4. Activate/Release of the outstanding interrupt on the impacted CPU
5. 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 LPI (if configured) or 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 interrupts (B) arrives so the GIC attempts to re-evaluate the CPU but fails to send the new interrupt as 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 as it becomes disabled or non-pending causing a CLEAR packet to be created.

3. The corresponding CPU group enabled 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-prioritisation 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:

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.

Workarounds

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:

1) Periodically toggle GICR_CTLR.DPG<x> to ensure that interrupts are delivered. The required frequency will be a function of system interrupt latency tolerance.

or

2) If not using LPIs then use GICD_I(S)C)ENBLERn 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)

SW should issue a DSB and toggle GICR_CTLR.DPG<x> after clearing the corresponding cpu group enable.

1717652

Wake_request may not be delivered if multiple cores are woken by PPIs at the same time

Description

The GIC is meant to raise a **wake_request** signal to the system control processor whenever there is a pending, enabled, and not active interrupt for a core.

In the PPI block, the two types of interrupt that generate wake requests are:

1. PPIs.
2. SGIs that were present before the core entered sleep mode. SGIs sent to sleeping cores are not impacted.

The PPI block sends each wake request as a packet to the GICD.

If multiple cores are waiting to send wake requests at the same time, by either transient backpressure on the bus to the GICD or by the credit being in use, then the destination core is incorrect and one of the cores might not be woken.

In small implementations, the window is a small number of clock cycles but may extend in distributed systems where the PPI block to GICD traffic is routed over general transport.

Implications

The implications are:

- If a system relies on a PPI (a core timer within the cluster) to wake a core from a power state where `GICR_WAKER.ProcessorSleep == 1`, then the PPI might not wake that core. LPIs, SPIs, and SGIs will still cause a wake.

Note:

- This issue is most likely to occur in systems that use a PPI as a general broadcast mechanism.
- Core power states such as WFI or WFE, where the core remains on are not affected.
- If a core has pending enabled SGIs when `GICR_WAKER.ProcessorSleep` is set, then the core might not receive an immediate wake request.

Workaround

When Secure software responds to a wake request to power up a core, it must:

1. Read the `GICR_PWRR` register to determine what other cores are on the same PPI block. Software can use the `GICR_PWRR.RDgroupOffset` value to determine the first core on the PPI block and then increment from there.
2. Read `GICR_ISPENDR*` for each core on the PPI block, to determine if the core must be woken.

3. Wake any other cores that it finds with pending PPIs or SGIs.

Note: To avoid spurious wakes, software might also check GICR_ISENBLE and GICR_ISACTIVER.

1494863

SPI recall failure without subsequent trigger

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 programming 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.

1196726

CLEAR command may continue after a SYNC

Description

A **SYNC** command indicates that all previous ITS commands have taken effect and that current ITS programming will apply to all new incoming GITS_TRANSLATER writes.

To limit the delay to software, the GIC acknowledges all commands as soon as possible.

Sometimes, under heavy load, when executing a **CLEAR** command followed by a **SYNC** command, it is possible for new interrupts to be cleared erroneously.

Implications

The GIC might clear the pending state of interrupts that arrive shortly after the **SYNC** command, for interrupts that match the previous **CLEAR** command.

Workaround

Insert an **INV** command between the **CLEAR** and **SYNC**

CLEAR
INV
SYNC

1118145

QACTIVE may not be driven HIGH on qreqn_its_<n> transition

Description

The errata does not impact monolithic configurations where the ITS and Distributor share a slave port.

The Distributor of the GIC-600 has a number of Q-Channels.

qreqn is used for hierarchically clock gating the clock to the Distributor.

qreqn_its_<n> are used for power control and disconnecting the interfaces between the Distributor and each ITS.

If a transition occurs on a **qreqn_its_<n>** while the main interface is in low power mode and the clock is stopped x cycles after the **qreqn_its_<n>**, the **qactive** may not be asserted. x is the depth of the implemented CDC synchronizer.

Implications

GIC will not respond to the **qreqn_its_<n>** transition until the clock is restarted and may not raise **qactive**.

Workaround

There are two workarounds:

1. Do not stop the clock while completing transitions on **qreqn_its_<n>**.
2. Write to GICD_FCTLR.Q_DENY to prevent the main Q-Channel from accepting entry requests until the ITS powerdown sequence is complete.

1109146

GICD_CTLR.RWP may be stuck at 1 if already 1 when rt_owner changes

Description

This issue only affects multi-chip configurations.

GIC-600 has a Routing Table (RT) that controls the partitioning of SPIs and chip address routing information. A single chip is defined as the owner of that table at any one time (the `rt_owner`).

The GIC supports moving that owner between chips so that the `rt_owner` is the last powered on chip.

If GICD_CTLR.RWP is 1 due to an SPI being recalled by the old `rt_owner` or by a GICD_CTLR update, when the `rt_owner` is changed then RWP may never drop.

Implications

RWP will remain high and will not drop causing poll timeouts.

Workarounds

There are 2 possible workarounds:

1. Never change the `rt_owner` and ensure that the first chip on is the last chip off.
2. Ensure that no processor can access the Distributor while the change is taking place and that `RWP == 0` before commencing the change.

1014611

Lower or same priority PPI/SGI may not be sent until after deactivate of previous interrupt

Description

If there are 2 pending enabled interrupts (PPIs or SGIs) for a CPU then if:

1. A search trigger occurs due to a control packet, deactivate or an edge on any **ppi<x>** wire
2. An activate occurs from the CPU.
3. The second highest priority interrupt may not be sent until the next trigger.

Higher priority interrupts will always be delivered.

Implication

PPIs and SGIs of the same or lower priority may not be delivered until after the deactivate of the activated interrupt.

Workaround

If you want to be certain of receiving PPI and SGIs of the same or lower priority then issue a deactivate to any non-active PPI or SGI on the same CPU with the same security group

996333

Disabled Secure SPIs blocking LPIs

Description

If the Secure CPU group enables, GICD_CTLR.EnableSG0 or GICD_CTLR.EnableSG1, are disabled while there are pending and enabled Secure SPIs waiting to be delivered to the CPU then the GIC will not accept an LPI into the slot unless its priority exceeds that of the Secure SPI.

If all 5 slots are in this state, then LPIs will not be delivered until an SPI (of any group) arrives and is serviced.

Implications

LPIs get stuck behind higher priority but disabled interrupts.

Workaround

Disable Secure SPIs using GICD_ICENABLER before disabling Secure CPU group enables, GICD_CTLR.EnableSG0 or GICD_CTLR.EnableSG1.

990914

Issues around PT triggering affecting LPI delivery

Description

Under certain conditions a search trigger event may be missed resulting in LPIs in the Pending Table for a CPU that falls outside of the search pointers not to be considered for delivery until the next trigger event occurs.

LPIs in the cache are not affected and will continue to be delivered. They will also cause trigger events which should retrigger the PT and restart the search.

Implications

Interrupts may be left in the PT until the next trigger event occurs.

Some additional odd behavior may be seen in LPI prioritization. However, once interrupts have spilled to the PT the latency of those interrupts will inevitably be increased due to additional latency of searching for and extracting those interrupts from the main memory.

Workaround

There are two parts to the workaround. The first significantly reduces the probability of hitting the preconditions for very low cost. The second ensures that if hit, the GIC will recover:

Part 1

Set GICD_FCTLR.AT (bit 24). This modifies how the TGT\$ triggers and will have a significant impact in reducing the probability of hitting the defect. The impact of setting this is minimal and Arm recommends it is always set.

Part2

The GIC can be poked to re-search on a regular interval. This could be done using either an external timer or by using the PMU to generate an SPI.

Periodically performing any of the trigger events described above will restart the search.

Note that this only needs to be done if any interrupts have been written to the PT and this can also be detected by using the PMU.

988378

Target cache presenting incorrect priorities when DS is set for filter

Description

In configurations where GICD_CTLR.DS is set then the priority is not correctly shifted when prioritizing LPIs.

Implications

Interrupt prioritization will not be as expected.

Workaround

Use priority 00 for LPIs

957913

Interrupt on Target_Cache Return interface can be missed by ITS commands

Description

When an ITS command occurs, it is supposed to impact all relevant LPIs in the system. However, there is a chance that an interrupt will be missed if the following conditions are met:

1. There are 3 unique LPIs for a particular PE in the Target Cache (TC).
Note for this to happen one of them must be highest priority interrupt for the PE.
2. A higher priority SPI arrives and is sent to the PE and evicts the LPI from the PE.

In these circumstances the TC returns one of the lowest priority LPIs back to the LPI cache.

If there is an ITS command that impacts the LPI being evicted by the TGT cache and the following events are also true:

1. There are no other interrupts for the PE in the cache.
2. There have been at least 4 other unique LPIs that do not hit in the LPI cache and are currently fetching properties as they missed in the LPI cache (that is, they all arrived within the last `t_memory_access` time).
3. The search of the cache (as required by the command) completes before an LFA frees up.

If all of the above conditions align, then the single interrupt on the TC Return path may be missed by the command.

Note that disabled interrupts cannot be affected by this errata so the action of enabling a disabled/masked interrupt via INV cannot fail.

Implications

The implications of missing the command are as follows:

CLR/DISCARD: The interrupt may remain pending on the TGT resulting in a single spurious interrupt to the current PE.

INV: The interrupt may maintain and cache its old properties meaning that future interrupts may remain enabled or take an old priority.

MOV: The interrupt may remain on the old target and not be moved.

Workaround

CLR/DISCARD: The spurious interrupt should be ignored.

INV: Repeating the INV will drastically reduce (but not remove) the probability of hitting this defect. Note that as enables will always work so interrupts should always be delivered eventually.

MOV/MOVALL: The only failsafe mechanism is to insert an INT command after the MOV. This will ensure that interrupts are never missed at the cost of a spurious interrupt.

907913

PT coarse map may be written back when invalid following sleep completion and wakeup in PT

Description

If GICD_CTLR is updated, in a short window as sleep is completing, then the coarse map is left in dirty state.

If a new interrupt arrives via PT set then the coarse map could be overwritten.

Implications

Loss of interrupts.

Workaround

Poll GICR_WAKER.Quiescent before changing GICD_CTLR.

If aborting sleep do INV + SYNC before enabling GICD_CTLR.

Category B (rare)

975678

Broadcast GEN_SGI generates interrupt for the source CPU when >10 CPUs

Description

Broadcast SGIs for sent from CPUs with MPIDR values of:

0.0.0.10

0.0.0.11

0.0.0.12

0.0.0.13

will send to themselves as well as all other CPUs.

Implications

A spurious SGI will be received.

Workaround

Ignore the spurious interrupt or issue targeted SGIs instead of the broadcast.

931156

ITS not ignoring corrupted RDATA when ITS receives an ERROR response

Description

If the ITS issues a read which is returned with an error from the memory system then it reports the error via the Distributor error registers.

It should also drop the relevant interrupt and allow software to take action based on the reported error data. Instead it uses the data for translation and if the corrupted data happens to map to a valid set of translation data then a spurious interrupt may be created.

Implications

If the corrupted data maps to valid data then it will be used in translation and potentially cached.

Workaround

If a memory access is reported via the GICD software error record, then software should flush the ITS caches via GITS_FCTLRL.

Category C

2023459

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

Description

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

Conditions

Target field [51:47] of the RDbase field are nonzero during a **MAPC, MOVALL, VMAPP, or VMOVP** command.

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.

2023457

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

Description

The upper 32 bits of the GICD_IROUTERn registers contain the Affinity3 field. Also, 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 zero rather than the programmed value.

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 two 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 owned according to the GICD_CHIPR registers.

1621321

SPI pipe does not always write back single-bit error corrections - including during scrub

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.

1451068

DCHIPR reads 0 from chips that are not the default owner

Description

The DCHIPR register contains 2 fields:

- RTOwner - Identifies the chip that is the current owner of the *Routing Table* (RT), also known as Default Owner or Crown Socket.
- PUP - Indicates that RTOwner is performing an update.

The only reason for writing this register is to change the RTOwner, to allow for the RTOwner Socket to be powered down.

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

Implications

The software receives an incorrect indication that PUP has dropped early.

If the software attempts to start a new access, for example chip offline, before PUP has completed, then the new access is rejected.

Rejected writes to update chip state are detectable, as the relevant CHIPR register do not record the updated value.

Workaround

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

1335020

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

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:

1. Do not enable the interrupts generated from error record 0 and ignore error record 0. (This is the reset behaviour of the GIC)
2. If the interrupts are enabled clear error record 0 after every remote SGI (based on receiving the error interrupt)
3. 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.

1109171

Possible SPI corruption if GICD_IERRR<n> written to 1 without being read first in multi-chip configs

Description

This does not impact single-chip configurations.

GIC-600 contains a register GICD_IERRR<n> which is involved in containment and correction of SPI after an uncorrectable RAM error occurs.

If an attempt is made to clear an error while the GIC is attempting to send the SPI to another chip, then the error clear may be successful when the SPI is not yet in a safe state.

This behavior only occurs if both the following conditions occur:

1. The error on the SPI occurs in the small window between deciding to send an SPI to another chip.
2. Software does not read GICD_IERRR<n> to observe the error before attempting to clear it using a GICR_IERRR<n> write.

Implications

Behavior of that SPI is UNPREDICTABLE.

Workaround

No software updates should be required because software should not attempt to clear errors not identified by reads to GICD_IERRR<n>.

1104459

ITS debug overflow not recorded in trace registers

Description

This only impacts configurations with an ITS.

If more than one translation or command error occurs in a single ITS within a very short time window, then the overflow bit may not be correctly recorded.

Implications

The ITS error record may not indicate an overflow when multiple errors are detected.

Workaround

None is required because software should be programming the GIC to prevent errors from occurring.

1098957

UNPREDICTABLE behavior on failed attempt to connect chips

Description

This does not impact single chip configurations.

GIC-600 has a connection process for connecting chips together under the control of Secure software.

If an attempt is made to connect to a chip in an unsuitable state the GIC is supposed to reject the connection and return to a known good state. However, the GIC fails to store the address to return the response and may respond to the incorrect address.

Unsuitable states include being already connected, out-of-range or GICD_CLTR values are incorrect.

Implications

Behavior is UNPREDICTABLE and can include any of:

1. Failure to send a response to the correct address.
2. Deadlock.
3. Loss of coherence between chips.

Workaround

No software change is expected as software should only ever be attempting to connect chips in the correct state.

1071611

DPG might allow a single 1ofN interrupt to be sent after being set

Description

A single 1ofN interrupt might ignore GICR_CTLR.DPG settings if the following events occur:

1. DPG is programmed while the CPU group enables are enabled.
2. There are multiple 1ofN SPIs that have been routed to CPUs that are processing other higher priority interrupts.

Implications

A GIC might deliver an unexpected but valid 1ofN SPI interrupt to a CPU that should not receive it.

Workaround

No workaround is expected to be required as the only impact is a spurious interrupt to a CPU that is able to accept it.

However, the issue can be prevented by only changing DPG when the CPU groups enables are off.

1049857

PMU filtering for Deactivate Event incorrect

Description

The PMU UP_DEACT Event is intended to allow tracking of SPI deactivates sent to the GIC from the CPUs.

The PMU will correctly count unfiltered deactivates, however, attempting to count deactivates filtered by CPU (or group of CPUs) uses incorrect CPU information and will give an invalid result.

Implications

Filtered PMU count of Deactivates will be incorrect.

Workaround

Use an unfiltered counter or count Deactivates via code added in the ISR.

1003544

PPI does not mask out GRPMOD when GICD_CTLR.DS=1

Description

This only impact systems with a single security state: GICD_CTLR.DS == 1

The PPI block considers GRPMOD when determining if the CPU group is enabled for a particular interrupt.

Implications

Interrupts with GRPMOD == 1 and GRP == 0 will not be delivered.

There are two mechanisms for GRPMOD getting set:

1. Programming of GICR_GRPMODR before GICD_CTLR.DS is set. This requires GICD_CTLR.DS to be configured as programmable.
2. Uncorrectable errors corrupting the GRPMOD bit in the PPI RAM cannot be corrected once DS is set. This requires corruption and will slightly decrease the MTBF.

Workaround

The workaround for 1) is to program GICD_GRPMODR to 0 before setting DS. 0 is the reset value of GICD_GRPMOD and there is no known use case for programming the GIC with 2 security levels and then disabling security.

If the RAM is corrupted with an uncorrectable error which results in GICD_GRPMODR == 1 for an interrupt, then there is no way to reprogram it and the only way to recover is to:

1. Reprogram and use the interrupt as a Group1 interrupt.
2. Reset the PPI block following powerdown sequence and reprogram.

963506

Deactivate - Activate ordering violation may leave interrupt not active

Description

If a Deactivate is issued over the GIC-Stream interface followed by an Activate to the same SGI or PPI within 3 GIC cycles then the Deactivate may be processed after the Activate.

This would leave the interrupt with Active = 0, meaning that it could be delivered again if it is pending or becomes pending.

Writes to GICR<x>_I(S)C)Active are not affected by this issue.

Interrupts cannot remain masked by this issue.

Software should never get into this state because under normal operation it should only deactivate interrupts that it has already activated and cannot, therefore, be activated again.

Implications

The SGI or PPI will remain inactive and may be delivered again. This could result in a spurious interrupt.

Workaround

A DSB after Deactivating an interrupt would avoid the issue but no workaround is believed to be necessary.

930020

IRQCR accesses trigger out of range SPI block software errors

Description

In configurations with < 8 SPI blocks (256 SPIs) then an OOR block error will be generated when GICP_IRQCR is programmed to internally route the PMU interrupt to an SPI.

and

In configurations with < 7 SPI blocks (224 SPIs) then an OOR block error will be generated when GICT_ERRIRQCR<n> is programmed to internally route the error handling or fault recovery interrupt to an SPI.

Implications

An OOR block error will be generated in RAS error record 0.

Workaround

Clear (and ignore) the incorrectly generated error after programming.