Arm Cortex-R52+ processor MP044

Software Developer Errata Notice

Date of issue: November 14, 2023

Non-Confidential

Document version: 6.0

Document ID: SDEN-2096952

This document contains all known errata since the r0p0 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**.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality status

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm Cortex-R52+ processor MP044, 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.

November 14, 2023: Changes in document version v6.0

ID	Status	Area	Category	Summary
2921272	New	Programmer	Category B	Executing LDM instruction might cause data corruption when HSCTLR.FI is set
3120632	Updated	Programmer	Category C	The debugger view of the number of cores might be incorrect when switching between Split and Lock modes
3120666	Updated	Programmer	Category C	Data ATB flush may not respond
3120670	Updated	Programmer	Category C	DTR flags not cleared on external debugger access while leaving Debug state

November 08, 2022: Changes in document version v5.0

ID	Status	Area	Category	Summary
2761868	New	Programmer	Category B	GICR_TYPER registers do not reflect affinity configuration inputs

July 27, 2022: Changes in document version v4.0

No new or updated errata in this document version.

March 23, 2022: Changes in document version v3.0

ID	Status	Area	Category	Summary
3120670	New	Programmer	Category C	DTR flags not cleared on external debugger access while leaving Debug state

April 20, 2021: Changes in document version v2.0

ID	Status	Area	Category	Summary
2124438	New	Programmer	Category A	VLDM/VSTM may corrupt data under certain conditions

March 26, 2021: Changes in document version v1.0

ID	Status	Area	Category	Summary
3120632	New	Programmer	Category C	The debugger view of the number of cores might be incorrect when switching between Split and Lock modes
3120666	New	Programmer	Category C	Data ATB flush may not respond
2074700	New	Programmer	Category C	ATB flush response may be delayed

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
2124438	Programmer	Category A	VLDM/VSTM may corrupt data under certain conditions	r0p0	r0p1
2761868	Programmer	Category B	GICR_TYPER registers do not reflect affinity configuration inputs	r0p0, r0p1	Open
2921272	Programmer	Category B	Executing LDM instruction might cause data corruption when HSCTLR.FI is set	r0p0, r0p1	Open
3120670	Programmer	Category C	DTR flags not cleared on external debugger access while leaving Debug state	r0p0, r0p1	Open
3120632	Programmer	Category C	The debugger view of the number of cores might be incorrect when switching between Split and Lock modes	r0p0, r0p1	Open
3120666	Programmer	Category C	Data ATB flush may not respond	r0p0, r0p1	Open
2074700	Programmer	Category C	ATB flush response may be delayed	r0p0	r0p1

Errata descriptions

Category A

2124438
VLDM/VSTM may corrupt data under certain conditions

Status

Affects: Cortex-R52+
Fault Type: Programmer, Category A
Fault Status: Present in r0p0. Fixed in r0p1.

Description

The Cortex-R52+ processor supports fetching of memory data in either big-endian or little-endian modes. Changing between the two can be enabled by executing the **`SETEND`** instruction.

The processor also supports instructions that load (or store) multiple registers to (or from) the Advanced SIMD and floating-point register file. These instructions are the **`VLDM`**, **`VSTM,`** and their aliases **`VPOP`**, **`FLDMDBX`**, **`FLDMIAX`**, **`VPUSH`**, **`FSTMDBX`**, **`FSTMIAX`**.

Under certain and very specific conditions, in a program that contains both a **`SETEND`** instruction that changes the current endianness and a variant of a **`VLDM`** (or **`VSTM`**) instruction, the processor corrupts the loaded (or stored) data.

Configurations affected

The erratum only affects systems where the program contains instructions that change the endianness between big-endian and little-endian.

Conditions

This erratum occurs when all the following conditions are met:

- The program executes a load/store instruction followed by a conditional branch instruction
- The load/store instruction and the conditional branch instruction are dual-issued. This happens when the CPUACTLR.DIDIS field is set to 0.
- The load/store instruction stalls in the pipeline for at least five cycles. A load that misses in the data cache typically stalls for at least five cycles.
- The conditional branch instruction mispredicts at least once in the execution
    - In the fail path of the mispredicted branch, the program contains a **`SETEND`** instruction within

the first five instructions

- In the correct path of the mispredicted branch, a `VLDM.64/VSTM.64` is the first executed instruction and this instruction operates on D registers but accesses a memory location whose base address is not 8-byte aligned.

Implications

If this erratum occurs, the processor corrupts the loaded (or stored) data.

Workaround

To avoid this erratum, insert an `ISB` instruction before the `SETEND` instruction.

Category A (rare)

There are no errata in this category.

Category B

2761868
GICR_TYPER registers do not reflect affinity configuration inputs

Status

Affects: Cortex-R52+
Fault Type: Programmer Category B
Fault Status: Present in r0p0, r0p1. Open

Description

The Cortex-R52+ processor has an identifier which is set via the processor top-level configuration inputs **CFGMPIDRAFF1** and **CFGMPIDRAFF2**. These are referred to as the Aff1 and Aff2 identifier fields. The Aff3 identifier field is always zero. These affinity fields are reported through the MPIDR system register for each core and the GICR_TYPER register for each associated Redistributor. Because of this erratum, the GICR_TYPER.Aff1 and GICR_TYPER.Aff2 fields are always zero.

Conditions

This erratum occurs if either of the **CFGMPIDRAFF1** or **CFGMPIDRAFF2** inputs is nonzero, and one of the GICR_TYPER registers are read.

Implications

An operating system running on a particular core might compare the affinity values of the core from the MPIDR with the affinity values found in the GICR_TYPER of the various Redistributors in order to identify which Redistributor is associated with the core. This erratum will cause all such comparisons to mismatch which might lead to a failure or hang during booting.

Workaround

Software which identifies Cortex-R52+ Redistributors by matching affinity values should compare the Aff0 value only and ignore the Aff1 and Aff2 values.

Note

The Redistributors' registers within a Cortex-R52+ processor cluster are only accessible to the cores within that cluster and all the Redistributors within a cluster are associated with cores within the cluster or the GIC export port for that cluster. Therefore, it is only necessary to compare the Aff0 values to precisely identify the correct Redistributor.

2921272
Executing LDM instruction might cause data corruption when HSCTLR.FI is set

Status

Affects: Cortex-R52+
Fault Type: Programmer Category B
Fault Status: Present in r0p0, r0p1. Open

Description

The Cortex-R52+ processor supports a variety of asynchronous exceptions, such as interrupts and System Error aborts. The processor can also be optionally configured with floating-point instructions support. Although Cortex-R52+ is an in-order processor, the floating-point divisions and square root operations complete out-of-order with respect to other instructions.

Because of this erratum, a rare combination of a floating-point division (or square root) instruction, a load-multiple instruction, and an asynchronous exception might cause the contents of one of the integer registers to be corrupted.

Configurations affected

This erratum affects configurations of the processor with FP_SIMD<n> set to either 1 or 3.

Conditions

This erratum occurs when both of the following are true:

- Out-of-order floating-point divisions and square roots are enabled (CPUACTLR.OOODIVDIS is cleared to 0). This is the reset value.
- Fast Interrupts are enabled (HSCTLR.FI is set to 1). This is not the reset value, and has to be explicitly programmed.

and the following sequence of conditions is met:

1. A VDIV or VSQRT instruction is executed.
2. An integer instruction is executed that produces <Rn>.
3. One of the next two instructions is a conditional, integer, load-multiple instruction which loads at least two registers and consumes <Rn> as its base register.
4. An asynchronous exception is taken.

Moreover, a very specific timing relationship among instructions (1), (2), (3) and exception (4) needs to be present for the erratum to occur.

Implications

When this erratum occurs, the base register of instruction (3) will be corrupted when exception (4) is taken. This implies that the instruction cannot be correctly replayed when the exception handler returns.

Workaround

You can avoid this erratum by doing either of the following:

- Disable out-of-order floating-point divisions and square roots, by setting CPUACTLR.OOODIVDIS to 1. There might be a minimal performance overhead. Indicative measurements show that the performance of several known benchmarks are impacted by less than 0.1%.
- Disable Fast Interrupts, by clearing HSCTLR.FI to 0. The processor interrupt latency might be affected, at the worst case by waiting for 128 bytes of memory to be read.

Category B (rare)

There are no errata in this category.

Category C

3120670
DTR flags not cleared on external debugger access while leaving Debug state

Status

Affects: Cortex-R52+
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Open.

Description

The *Data Transfer Registers* (DTRs) provide a mechanism to transfer data between an external debugger and the core. They consist of write-only registers to transmit data (DBGDTRTX_EL0 and DBGDTRTXint), read-only registers to receive data (DBGDTRRX_EL0 and DBGDTRRXint), and associated data control flags.

Due to this erratum, if these registers are accessed by the external debugger while the debug exit procedure is in progress, then the accesses will go ahead but the flow control flags will not be updated correctly.

Configurations affected

This erratum affects all configurations.

Conditions

This erratum occurs when the following sequence of conditions is met:

1. The debugger requests a debug exit.
2. The debugger does an external access to the DBGDTRRX/ DBGDTRTX, while the debug exit is ongoing.
3. Certain microarchitectural timing conditions are met.

Implications

For an external read to DBGDTRTX, the EDSCR.TXU and EDSCR.TXfull flags will not be updated.
For an external write to DBGDTRRX, the write will be ignored and the EDSCR.RXO and EDSCR.RXfull flags will not be updated.

Workaround

There is no workaround.

Please note that this erratum is now published as EN ID 3120670. The previous EN ID 2289296 is deprecated. This is done to work around a document generation issue.

3120632 The debugger view of the number of cores might be incorrect when switching between Split and Lock modes

Status

Affects: Cortex-R52+
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Open.

Description

The Cortex-R52+ processor can be configured with Split-Lock, which allows the number of cores that the processor presents to vary, under Cold reset, between Split mode and Lock mode. The debugger uses the entries in the Cortex-R52+ debug ROM table to determine the number and offsets of the cores in the processor. The processor ROM table always indicates the current number of cores because of this erratum. If a debugger reads the ROM table and the mode is subsequently switched, the information that it reads is incorrect.

Configurations Affected

This erratum affects the Cortex-R52+ processor when configured with Split-Lock.

Conditions

1. The SoC dynamically switches the processor between Split and Lock mode under Cold reset.
2. The SoC does not reset its debug logic when resetting the processor.
3. The debugger performs debug accesses.

Implications

Debug accesses might not work as expected because:

- A debugger connected to a part that boots up in Lock mode and switches to Split mode is not aware of all the accessible cores.
- A debugger connected to a part that boots up in Split mode and switches to Lock mode might attempt to access cores that are now redundant. Such accesses receive an error response.

Workaround

To ensure a debugger has a complete description of the system, the ROM table must be read following each reset of all cores in the Cortex-R52 processor. A reset of a core can be detected by reading EDPRSR.

Please note that this erratum is now published as EN ID 3120632. The previous EN ID 2071697 is deprecated. This is done to work around a document generation issue.

3120666
Data ATB flush may not respond

Status

Affects: Cortex-R52+
Fault Type: Programmer Category C
Fault Status: Present in r0p0 and r0p1. Open.

Description:

The *Embedded Trace Macrocell* (ETM) supports an external flush request for each ATB bus. If the ETM is active and configured with no data trace, there will be no flush response on the data ATB.

Configurations affected

This erratum affects all processor configurations.

Conditions

This erratum occurs when all of the following conditions are met:

- The ETM is enabled
- Data trace is disabled (both TRCCONFIGR.DA and TRCCONFIGR.DV are clear)
- The ETM is not in a low-power state due to WFI or WFE
- An ATB flush is requested on the data ATB bus

Implications

External trace infrastructure which is waiting for trace to be captured may stall forever (for example in a "flush and stop" scenario). Instruction trace will flush as expected, but this may not be observed at the trace capture device. Disabling the ETM or entering a low-power state will restore the expected behavior.

Workaround

To avoid this erratum:

- Enable TRCCONFIGR.DA or TRCCONFIGR.DV
- Program all of the following registers to zero if data trace is not required:
    - TRCVDARCCTLR
    - TRCVDCTLR
    - TRCVDSACCTLR

  - TRCEVENTCTL1R.DATAEN

This will slightly increase power consumption, but it will not cause any data trace generation.

Please note that this erratum is now published as EN ID 3120666. The previous EN ID 2048344 is deprecated. This is done to work around a document generation issue.

2074700
ATB flush response may be delayed

Status

Affects: Cortex-R52+
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

Description:

The *Embedded Trace Macrocell* (ETM) supports an external flush request for each ATB bus. Under certain timing conditions, an AFREADY response from the Cortex-R52+ processor may be delayed until a new ATB transfer is generated by the Cortex-R52+ processor.

Configurations affected

This erratum affects all Cortex-R52+ processor configurations.

Conditions

The erratum occurs if the following sequence of conditions are met:

1. The ETM is enabled
2. Trace data is generated on the ATB bus
3. An external flush request is generated
4. Trace data stops being generated due to filtering in the ETM or the ETM being disabled

Implications

External trace infrastructure which is waiting for trace to be captured may stall forever (for example in a 'flush and stop' scenario). All of the trace data will be captured, but it is not possible to identify this by polling the trace capture device. If the flush is acknowledged, this can be treated as a reliable indication.

If the ETM is enabled again after the erratum has been triggered, the flush logic should become active again. If a flush is generated while trace is being generated, the only effect will be a delay in acknowledging the flush. This should not have any observable impact.

In a system with multiple trace sources, the delayed flush response may prevent other trace sources from accessing the ATB bus if they are generating trace while the flush is in progress. This could cause trace to be lost from these other sources.

Workaround

There is no workaround to reliably avoid this erratum. As an alternative to waiting for the flush to be acknowledged, a sufficiently long timeout can be used if it is likely that trace generation has stopped.