



This document contains all known errata since the rOp0 release of the product.

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

Date of issue: 20-Jan-2022

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 © 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 Morello, 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	5
Scope	5
Categorization of errata	5
Change Control	6
Errata summary table	7
Errata descriptions	8
Category A	8
Category A (rare)	8
Category B	9
2430082 System Generic Timer - CNTFRQ register in CNTBaseN and CNTCTLBase views inconsistent	9
2430086 Client mode tag updates can be lost under certain conditions	11
2166107 AP[2] bit (dirty bit) may be updated even when store fails due to capability bounds fault	12
2066137 Incorrect target Exception level on LC fault	13
Category B (rare)	13
Category C	14
2101099 Trapped capability bounds faults not always trapped	14

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.

20-Jan-2022: Changes in document version v1.0

ID	Status	Area	Category	Summary
2066137	New	Programmer	Category B	Incorrect target Exception level on LC fault.
2166107	New	Programmer	Category B	AP[2] bit (dirty bit) may be updated even when store fails due to capability bounds fault
2430082	New	Programmer	Category B	Generic Timer - CNTFRQ architecture alignment issue
2430086	New	Programmer	Category B	Client mode tag updates can be lost under certain circumstances
2101099	New	Programmer	Category C	Trapped capability bounds faults not always trapped

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
2430082	Programmer	Category B	Generic Timer - CNTFRQ architecture alignment issue	r0p0, r0p1	Open
2430086	Programmer	Category B	Client mode tag updates can be lost under certain circumstances	r0p0, r0p1	Open
2166107	Programmer	Category B	AP[2] bit (dirty bit) may be updated even when store fails due to capability bounds fault	r0p0, r0p1	Open
2066137	Programmer	Category B	Incorrect target Exception level on LC fault.	r0p0, r0p1	Open
2101099	Programmer	Category C	Trapped capability bounds faults not always trapped	r0p0, r0p1	Open

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

2430082

System Generic Timer - CNTFRQ register in CNTBaseN and CNTCTLBase views inconsistent

Status

Affects: Morello TAP, Corstone-700

Fault Type: Programmer Cat B

Fault Status: Present in r0p0, r0p1

Description

The Generic Timer register CNTFRQ behavior has been updated for Armv8-A.

For the current Armv7-A implementation, the register is visible in two frames CNTBaseN and CNTCTLBase which have been implemented as independent registers.

Software, which expects the Armv8-A behavior, writes the expected value to the CNTFRQ register in the CNTCTLBase frame. This value is then expected to be reflected when read from the CNTFRQ register in the CNTBaseN frame.

However, as these registers are independent the values are not reflected.

Configurations affected

All configurations are affected.

Conditions

Software writes a value to the CNTFRQ register through the CNTCTLBase frame.

Software reads the CNTFRQ register through the CNTBaseN frame.

The value of CNTFRQ read through the CNTBaseN frame does not reflect the value written through the CNTCTLBase frame as these are implemented as independent registers.

Implications

OS software may fail to boot due to inconsistencies in the CNTFRQ views.

Workaround

In the current Armv7-A implementation although the CNTFRQ is 'read-only', for initial configuration it can be written through the CNTBaseN frame.

Software therefore must write the required CNTFRQ value to both the CNTBaseN and CNTCTLBase frames.

This workaround ensures consistency when reading the CNTFRQ value from either CNTBaseN or CNTCTLBase frames.

For example:

```
mmio_write_32(ARM_SYS_TIMCTL_BASE + CNTCTLBASE_CNTFRQ, freq_val);
```

```
mmio_write_32(ARM_SYS_CNT_BASE_NS + CNTBASEN_CNTFRQ, freq_val);
```

2430086

Client mode tag updates can be lost under certain conditions

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r0p1

Description

In Client mode, the tag bit that indicates a valid capability may not be updated properly under certain circumstances.

Configurations

Configurations which use Client mode are affected.

Conditions

When the system is configured in Client mode, under certain microarchitectural conditions an operation to set or clear a tag in memory may not happen. This results in one or more tags being in a stale (incorrect) state.

Implications

Pointers can have the wrong value of the tag bit. This means that a tag can unexpectedly be 0 when it is supposed to be 1, or 1 when it is supposed to be 0. The first can result in an unexpected exception. The second can result in a violation of the security guarantees of the Morello architecture.

Workaround

Use only Server mode for most testing, particularly when the Morello architecture is being assessed.

Note: It may be possible to use Client mode for certain kinds of testing where the issue described in this erratum is not important. For example, it may be possible to do some performance testing. For this type of use, we recommend that you work closely with Arm.

2166107

AP[2] bit (dirty bit) may be updated even when store fails due to capability bounds fault

Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r0p1

Description

When an unaligned store using a non-capability pointer takes a capability bounds fault while the Dirty Bit Modifier (DBM) is enabled, the AP[2] bit may be updated erroneously.

The correct behavior is to leave the AP[2] bit unchanged.

This erratum causes the AP[2] bit to be updated under the stated conditions.

Configurations affected

All configurations are affected.

Conditions

This erratum occurs when an unaligned store using a non-capability pointer causes a bounds fault while DBM is enabled.

Implications

If the stated conditions are met, then AP[2] is updated rather than being left unmodified.

Note: If there is no bounds fault, it would be correct to update AP[2].

Workaround

Where software cannot handle AP[2] being updated in this case, it should not enable DBM, and manage AP[2] in software.

2066137

Incorrect target Exception level on LC fault

Status

Fault Type: Programmer Category B
Fault Status: Present in rOp0, rOp1

Description

When SCR_EL3.EA = 1 then LC faults behave as if they are synchronous External aborts and are routed to EL3. If SCR_EL3.EA = 0 and HCR_EL2.TEA = 1, they are routed to EL2.

The correct behavior is for LC faults to behave as stage 1/2 Data aborts.

This erratum can cause LC faults to be routed to the wrong Exception level.

Configurations

All configurations.

Conditions

This incorrect fault routing behavior can occur when SCR_EL3.EA is set (or if HCR_EL2.TEA is set).

Implications

The incorrect exception routing for LC faults can cause correctly written software to behave incorrectly.

Workaround

The workaround is to require SCR_EL3.EA = HCR_EL2.TEA = 0. It is worth noting that it would be acceptable to set SCR_EL3.EA while running (only) in EL3.

Category B (rare)

There are no errata in this category.

Category C

2101099

Trapped capability bounds faults not always trapped

Status

Fault Type: Programmer Category C

Fault Status: Present in rOp0, rOp1

Description

When access to Morello functionality is trapped to EL3 by CPTR_EL3.EC, or to EL2 by CPTR_EL2.TC, and executing at an Exception level below the target Exception level, an unaligned non-capability store that is split across a page boundary and takes a bounds fault is treated as not trapped by those controls. The correct behavior is to take the exception at the lowest Exception level trapping the access. This erratum can route bounds fault exceptions created under these conditions to the wrong Exception level.

Configurations

All configurations are affected.

Conditions

This incorrect behavior requires:

- Access to capability functionality to be trapped to EL3 by CPTR_EL3.EC or to EL2 by CPTR_EL2.TC
- Execution at an Exception level below the trapping level
- An unaligned store using a non-capability pointer that is split across a page boundary and that takes a bounds fault.

Implications

Incorrect exception routing can occur under the stated conditions. Arm expects that a higher Exception level sets the maximum PCC and DDC for lower Exception levels where access to Morello is disabled and so LC faults will not be generated, so no workaround is necessary.

Workaround

No workaround is necessary.

