



## Arm Cortex-A76 (MP052) Software Developer Errata Notice

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, 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 <http://www.arm.com/company/policies/trademarks>.

Copyright © 2019 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.

## Product Status

The information in this document is for a product in development and is not final.

## Web address

<http://www.arm.com/>.

## Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

## Feedback on this document

If you have comments on content then send an e-mail to [errata@arm.com](mailto:errata@arm.com) giving:

- The document title.
- The document number: SDEN-885749.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

# Contents

---

<i>INTRODUCTION</i>	8	
<i>ERRATA SUMMARY TABLE</i>	16	
1315703	Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-write ordering violation	21
905797	Failure to enforce read-after-read ordering rules	22
961148	Reads from DSU CLUSTER* or ERX* system registers might return corrupted data	23
977072	Accessing certain Debug or Generic Timer system registers in AArch32 might cause incorrect system register values	24
981980	Interrupt is taken immediately after MSR DAIF instruction masks the interrupt	25
1043202	AArch32 T32 CLREX in an IT block will clear exclusive monitor even if it fails condition code check	26
1073348	Concurrent instruction TLB miss and mispredicted return instruction might fetch wrong instruction stream	27
1130799	TLBI VAAE1 or TLBI VAALE1 targeting a page within hardware page aggregated address translation data in the L2 TLB might cause corruption of address translation data	28
1165347	Continuous failing STREX because of another core snooping from speculatively executed atomic behind constantly mispredicted branch might cause livelock	29
1165522	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation	30
1188873	MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result	31
1207823	The exclusive monitor might end up tracking an incorrect cache line in the presence of a VA-alias, causing a false pass on the exclusive access sequence	32
1220197	Streaming store under specific conditions might cause deadlock or data corruption	33
1257314	Multiple floating-point divides/square roots concurrently completing back-to-back and flushing back-to-back might cause data corruption	34
1262606	Concurrent instruction TLB miss and mispredicted branch instruction located at the end of 32MB region might fetch wrong instruction stream	35
1262888	Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation	36
1275112	A T32 instruction inside an IT block followed by a mispredicted speculative instruction stream might cause a deadlock	37
1463225	Software Step might prevent interrupt recognition	38
1286807	Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-read ordering violation	39
1418040	MRRC reads of some Generic Timer system registers in AArch32 mode might return corrupt data	40
901865	Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock	40
930203	Persistent error response to transactions issued on behalf of Page descriptor Access bit and Dirty bit updates might livelock	42
930239	Failure to report or incorrect reporting of L2 data RAM ECC errors	43
933092	Critical beat data for an L2 cache miss, poisoned or tagged with error, consumed by a load without reporting an abort	44
933779	DBGDTRTX register fails to hold value through Warm reset	45
934968	DCPSx instruction with SCTLR_EL1.IESB = 1 while in debug state might not execute correctly	46

944783	Address breakpoint might cause a deadlock with certain AArch32 T32 code sequences	47
973981	L2 might report multiple RAS errors for the same prefetch request	48
974001	Deferred errors might cause silent data corruption following a hardware update of Access and Dirty bits in a translation table entry	49
978245	Executing unallocated encoding in conversion between floating-point and integer instruction class does not generate Undefined Instruction exception	50
980456	Stuck-at-fault in L1 instruction cache data array might cause deadlock with certain AArch32 T32 code sequences	51
986709	MRS to DBGDTR_EL0 might cause EDSCR.RXfull bit to clear incorrectly	52
988575	Unaligned cache line split load to NC or Device memory, tagged with poison or external error on its first half, might cause data corruption	53
1051464	CTI trigger occurring on same cycle PREADYCD is received might cause CTI trigger to be missed	54
1057923	Extra instruction might be executed during Halting Step when stepping WFI, WFE and some self-synchronizing system register writes	55
1069401	Debug APB accesses to the ELA RAM might return incorrect data	56
1085091	The ERXADDR_EL1 register might report an incorrect physical address for an L1 data tag RAM single-bit correctable ECC error	57
1096402	Exception packet for return stack match might return incorrect [E1:E0] field	58
1109624	Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock	59
1119735	16-bit T32 instruction close to breakpoint location might cause early breakpoint exception	60
1126105	Read from L1 instruction cache data array using RAMINDEX operation might return data from the wrong location	61
1144394	Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error	62
1145826	ERR0MISC0 might report incorrect BANK and SUBBANK values for parity errors in L1 instruction cache data array	63
1192279	IMPLEMENTATION DEFINED fault for unsupported atomic operations is not routed to proper Exception level	64
1214504	Direct access to L1 data TLB might report incorrect value of valid bit of the corresponding TLB entry	65
1220808	ERR0STATUS.SERR encoding is incorrect for error responses from slave and deferred data errors from slave which are not supported	66
1227053	Streaming writes to memory mapped Non-shareable and write-back might cause data corruption because of reordering	67
1244984	Illegal return event might corrupt PSTATE.UA0	68
1256788	Halting step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error	69
1264383	Write-Back load after two Device-nG* stores to the same physical address might get invalid data	70
1346756	TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0	71
1349291	Uncontainable (UC) SError might be incorrectly logged as an Unrecoverable (UEU) SError	72
1356341	L1D_CACHE access related PMU events and L1D_TLB access related PMU events increment on instructions/micro-operations excluded from these events	73
1395332	Read from PMCCNTR in AArch32 might return corrupted data	74
1406411	MSR DSPSR_EL0 while in debug state might not correctly update PSTATE.{N,C,Z,V,GE} on debug exit	75

---

<a href="#">1408724</a>	Portions of the branch target address recorded in ETM trace information might be incorrect for some branches immediately preceding an indirect branch with a malformed branch target address	76
<a href="#">1415323</a>	Ordering violation might occur when a load encounters an L1 tag RAM single bit ECC error when a snoop request targets the same line	77
<a href="#">1430754</a>	Write to External Debug Registers might cause a deadlock with certain AArch32 T32 code sequences	78
<a href="#">1487185</a>	Waypoints from previous session might cause single-shot comparator match when trace enabled	79
<a href="#">1490853</a>	TRCIDR3.CCITMIN value is incorrect	80

### r1p0 implementation fixes

Note the following errata might be fixed in some implementations of r1p0. This can be determined by reading the REVIDR\_EL1 register where a set bit indicates that the erratum is fixed in this part.

REVIDR_EL1[0]	1043202 AArch32 T32 CLREX in an IT block will clear exclusive monitor even if it fails condition code check
---------------	---

Note that there is no change to the MIDR\_EL1 which remains at r1p0 but the REVIDR\_EL1 is updated to indicate which errata are corrected. Software will identify this release through the combination of MIDR\_EL1 and REVIDR\_EL1.

### r2p0 implementation fixes

Note the following errata might be fixed in some implementations of r2p0. This can be determined by reading the REVIDR\_EL1 register where a set bit indicates that the erratum is fixed in this part.

REVIDR_EL1[0]	1220197 Streaming store under specific conditions might cause deadlock or data corruption
---------------	---

Note that there is no change to the MIDR\_EL1 which remains at r2p0 but the REVIDR\_EL1 is updated to indicate which errata are corrected. Software will identify this release through the combination of MIDR\_EL1 and REVIDR\_EL1.

### r3p0 implementation fixes

Note the following errata might be fixed in some implementations of r3p0. This can be determined by reading the REVIDR\_EL1 register where a set bit indicates that the erratum is fixed in this part.

REVIDR_EL1[1]	1315703 Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-write ordering violation
REVIDR_EL1[5]	1463225 Software Step might prevent interrupt recognition

Note that there is no change to the MIDR\_EL1 which remains at r3p0 but the REVIDR\_EL1 is updated to indicate which errata are corrected. Software will identify this release through the combination of MIDR\_EL1 and REVIDR\_EL1.

### r3p1 implementation fixes

Note the following errata might be fixed in some implementations of r3p1. This can be determined by reading the REVIDR\_EL1 register where a set bit indicates that the erratum is fixed in this part.

REVIDR_EL1[7]	1463225 Software Step might prevent interrupt recognition
---------------	---

Note that there is no change to the MIDR\_EL1 which remains at r3p1 but the REVIDR\_EL1 is updated to indicate which errata are corrected. Software will identify this release through the combination of MIDR\_EL1 and REVIDR\_EL1.

# 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 on page 16 identifies errata that have been fixed in each product revision.

<b>28-Jun-2019: Changes in document version 16.0</b>				
ID	Status	Area	Cat	Summary of erratum
1346756	Updated	Programmer	CatC	TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0
1487185	New	Programmer	CatC	Waypoints from previous session might cause single-shot comparator match when trace enabled
1490853	New	Programmer	CatC	TRCIDR3.CCITMIN value is incorrect

<b>30-Apr-2019: Changes in document version 15.0</b>				
ID	Status	Area	Cat	Summary of erratum
1463225	New	Programmer	CatB	Software Step might prevent interrupt recognition
1418040	Updated	Programmer	CatB (rare)	MRRC reads of some Generic Timer system registers in AArch32 mode might return corrupt data
1349291	Updated	Programmer	CatC	Uncontainable (UC) SError might be incorrectly logged as an Unrecoverable (UEU) SError
1356341	Updated	Programmer	CatC	L1D_CACHE access related PMU events and L1D_TLB access related PMU events increment on instructions/micro-operations excluded from these events
1395332	Updated	Programmer	CatC	Read from PMCCNTR in AArch32 might return corrupted data
1406411	Updated	Programmer	CatC	MSR DSPSR_EL0 while in debug state might not correctly update PSTATE.{N,C,Z,V,GE} on debug exit
1408724	Updated	Programmer	CatC	Portions of the branch target address recorded in ETM trace information might be incorrect for some branches immediately preceding an indirect branch with a malformed branch target address
1415323	Updated	Programmer	CatC	Ordering violation might occur when a load encounters an L1 tag RAM single bit ECC error when a snoop request targets the same line
1430754	New	Programmer	CatC	Write to External Debug Registers might cause a deadlock with certain AArch32 T32 code sequences

<b>26-Mar-2019: Changes in document version 14.0</b>				
ID	Status	Area	Cat	Summary of erratum

1418040	New	Programmer	CatB (rare)	MRRC reads of some Generic Timer system registers in AArch32 mode might return corrupt data
1406411	New	Programmer	CatC	MSR DSPSR_EL0 while in debug state might not correctly update PSTATE.{N,C,Z,V,GE} on debug exit
1408724	New	Programmer	CatC	Portions of the branch target address recorded in ETM trace information might be incorrect for some branches immediately preceding an indirect branch with a malformed branch target address
1415323	New	Programmer	CatC	Ordering violation might occur when a load encounters an L1 tag RAM single bit ECC error when a snoop request targets the same line

**08-Mar-2019: Changes in document version 13.0**

ID	Status	Area	Cat	Summary of erratum
1346756	New	Programmer	CatC	TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0
1349291	New	Programmer	CatC	Uncontainable (UC) SError might be incorrectly logged as an Unrecoverable (UEU) SError
1356341	New	Programmer	CatC	L1D_CACHE access related PMU events and L1D_TLB access related PMU events increment on instructions/micro-operations excluded from these events
1395332	New	Programmer	CatC	Read from PMCCNTR in AArch32 might return corrupted data

**21-Nov-2018: Changes in document version 12.0**

ID	Status	Area	Cat	Summary of erratum
1315703	New	Programmer	CatA (rare)	Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-write ordering violation
1257314	Updated	Programmer	CatB	Multiple floating-point divides/square roots concurrently completing back-to-back and flushing back-to-back might cause data corruption
1262606	Updated	Programmer	CatB	Concurrent instruction TLB miss and mispredicted branch instruction located at the end of 32MB region might fetch wrong instruction stream
1262888	Updated	Programmer	CatB	Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation
1275112	Updated	Programmer	CatB	A T32 instruction inside an IT block followed by a mispredicted speculative instruction stream might cause a deadlock

1286807	New	Programmer	CatB (rare)	Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-read ordering violation
1227053	Updated	Programmer	CatC	Streaming writes to memory mapped Non-shareable and write-back might cause data corruption because of reordering
1244984	Updated	Programmer	CatC	Illegal return event might corrupt PSTATE.UA0
1256788	Updated	Programmer	CatC	Halting step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error
1264383	Updated	Programmer	CatC	Write-Back load after two Device-nG* stores to the same physical address might get invalid data

**04-Oct-2018: Changes in document version 11.0**

ID	Status	Area	Cat	Summary of erratum
1257314	New	Programmer	CatB	Multiple floating-point divides/square roots concurrently completing back-to-back and flushing back-to-back might cause data corruption
1262606	New	Programmer	CatB	Concurrent instruction TLB miss and mispredicted branch instruction located at the end of 32MB region might fetch wrong instruction stream
1262888	New	Programmer	CatB	Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation
1275112	New	Programmer	CatB	A T32 instruction inside an IT block followed by a mispredicted speculative instruction stream might cause a deadlock
1227053	New	Programmer	CatC	Streaming writes to memory mapped Non-shareable and write-back might cause data corruption because of reordering
1244984	New	Programmer	CatC	Illegal return event might corrupt PSTATE.UA0
1256788	New	Programmer	CatC	Halting step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error
1264383	New	Programmer	CatC	Write-Back load after two Device-nG* stores to the same physical address might get invalid data

**07-Sep-2018: Changes in document version 10.0**

ID	Status	Area	Cat	Summary of erratum
1165522	Updated	Programmer	CatB	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation

1188873	Updated	Programmer	CatB	MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result
1220197	New	Programmer	CatB	Streaming store under specific conditions might cause deadlock or data corruption

**01-Aug-2018: Changes in document version 9.0**

ID	Status	Area	Cat	Summary of erratum
1130799	New	Programmer	CatB	TLBI VAAE1 or TLBI VAALE1 targeting a page within hardware page aggregated address translation data in the L2 TLB might cause corruption of address translation data
1165347	New	Programmer	CatB	Continuous failing STREX because of another core snooping from speculatively executed atomic behind constantly mispredicted branch might cause livelock
1165522	New	Programmer	CatB	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation
1188873	New	Programmer	CatB	MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result
1207823	New	Programmer	CatB	The exclusive monitor might end up tracking an incorrect cache line in the presence of a VA-alias, causing a false pass on the exclusive access sequence
1085091	Updated	Programmer	CatC	The ERXADDR_EL1 register might report an incorrect physical address for an L1 data tag RAM single-bit correctable ECC error
1096402	Updated	Programmer	CatC	Exception packet for return stack match might return incorrect [E1:E0] field
1109624	Updated	Programmer	CatC	Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock
1119735	Updated	Programmer	CatC	16-bit T32 instruction close to breakpoint location might cause early breakpoint exception
1126105	Updated	Programmer	CatC	Read from L1 instruction cache data array using RAMINDEX operation might return data from the wrong location
1144394	Updated	Programmer	CatC	Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error
1145826	Updated	Programmer	CatC	ERR0MISC0 might report incorrect BANK and SUBBANK values for parity errors in L1 instruction cache data array

1192279	New	Programmer	CatC	IMPLEMENTATION DEFINED fault for unsupported atomic operations is not routed to proper Exception level
1214504	New	Programmer	CatC	Direct access to L1 data TLB might report incorrect value of valid bit of the corresponding TLB entry
1220808	New	Programmer	CatC	ERRORSTATUS.SERR encoding is incorrect for error responses from slave and deferred data errors from slave which are not supported

**22-Jun-2018: Changes in document version 8.0**

ID	Status	Area	Cat	Summary of erratum
1085091	New	Programmer	CatC	The ERXADDR_EL1 register might report an incorrect physical address for an L1 data tag RAM single-bit correctable ECC error
1096402	New	Programmer	CatC	Exception packet for return stack match might return incorrect [E1:E0] field
1109624	New	Programmer	CatC	Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock
1119735	New	Programmer	CatC	16-bit T32 instruction close to breakpoint location might cause early breakpoint exception
1126105	New	Programmer	CatC	Read from L1 instruction cache data array using RAMINDEX operation might return data from the wrong location
1144394	New	Programmer	CatC	Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error
1145826	New	Programmer	CatC	ERR0MISC0 might report incorrect BANK and SUBBANK values for parity errors in L1 instruction cache data array

**13-Apr-2018: Changes in document version 7.0**

ID	Status	Area	Cat	Summary of erratum
1043202	Updated	Programmer	CatB	AArch32 T32 CLREX in an IT block will clear exclusive monitor even if it fails condition code check
1073348	New	Programmer	CatB	Concurrent instruction TLB miss and mispredicted return instruction might fetch wrong instruction stream
1051464	Updated	Programmer	CatC	CTI trigger occurring on same cycle PREADYCD is received might cause CTI trigger to be missed
1057923	New	Programmer	CatC	Extra instruction might be executed during Halting Step when stepping WFI, WFE and some self-synchronizing system register writes

1069401	New	Programmer	CatC	Debug APB accesses to the ELA RAM might return incorrect data
---------	-----	------------	------	---

**03-Apr-2018: Changes in document version 6.0**

ID	Status	Area	Cat	Summary of erratum
No new or updated errata in this document version.				

**09-Feb-2018: Changes in document version 5.0**

ID	Status	Area	Cat	Summary of erratum
1043202	New	Programmer	CatB	AArch32 T32 CLREX in an IT block will clear exclusive monitor even if it fails condition code check
1051464	New	Programmer	CatC	CTI trigger occurring on same cycle PREADYCD is received might cause CTI trigger to be missed

**23-Oct-2017: Changes in document version 4.0**

ID	Status	Area	Cat	Summary of erratum
977072	New	Programmer	CatB	Accessing certain Debug or Generic Timer system registers in AArch32 might cause incorrect system register values
981980	New	Programmer	CatB	Interrupt is taken immediately after MSR DAIF instruction masks the interrupt
944783	New	Programmer	CatC	Address breakpoint might cause a deadlock with certain AArch32 T32 code sequences
973981	New	Programmer	CatC	L2 might report multiple RAS errors for the same prefetch request
974001	New	Programmer	CatC	Deferred errors might cause silent data corruption following a hardware update of Access and Dirty bits in a translation table entry
978245	New	Programmer	CatC	Executing unallocated encoding in conversion between floating-point and integer instruction class does not generate Undefined Instruction exception
980456	New	Programmer	CatC	Stuck-at-fault in L1 instruction cache data array might cause deadlock with certain AArch32 T32 code sequences
986709	New	Programmer	CatC	MRS to DBGDTR_EL0 might cause EDSCR.RXfull bit to clear incorrectly
988575	New	Programmer	CatC	Unaligned cache line split load to NC or Device memory, tagged with poison or external error on its first half, might cause data corruption

**25-Sep-2017: Changes in document version 3.0**

ID	Status	Area	Cat	Summary of erratum
961148	New	Programmer	CatB	Reads from DSU CLUSTER* or ERX* system registers might return corrupted data
933092	New	Programmer	CatC	Critical beat data for an L2 cache miss, poisoned or tagged with error, consumed by a load without reporting an abort
933779	New	Programmer	CatC	DBGDTRTX register fails to hold value through Warm reset
934968	New	Programmer	CatC	DCPSx instruction with SCTLR_EL1.IESB = 1 while in debug state might not execute correctly

**18-Aug-2017: Changes in document version 2.0**

ID	Status	Area	Cat	Summary of erratum
905797	New	Programmer	CatB	Failure to enforce read-after-read ordering rules
901865	New	Programmer	CatC	Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock
930203	New	Programmer	CatC	Persistent error response to transactions issued on behalf of Page descriptor Access bit and Dirty bit updates might livelock
930239	New	Programmer	CatC	Failure to report or incorrect reporting of L2 data RAM ECC errors

**30-May-2017: Changes in document version 1.0**

ID	Status	Area	Cat	Summary of erratum
No errata in this document version.				

## Errata summary table

The errata associated with this product affect product versions as below.

ID	Cat	Summary	Found in versions	Fixed in version
1315703	CatA (rare)	Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-write ordering violation	r0p0, r1p0, r2p0, r3p0	r3p1
905797	CatB	Failure to enforce read-after-read ordering rules	r0p0	r1p0
961148	CatB	Reads from DSU CLUSTER* or ERX* system registers might return corrupted data	r0p0	r1p0
977072	CatB	Accessing certain Debug or Generic Timer system registers in AArch32 might cause incorrect system register values	r0p0	r1p0
981980	CatB	Interrupt is taken immediately after MSR DAIF instruction masks the interrupt	r0p0	r1p0
1043202	CatB	AArch32 T32 CLREX in an IT block will clear exclusive monitor even if it fails condition code check	r0p0, r1p0	r2p0
1073348	CatB	Concurrent instruction TLB miss and mispredicted return instruction might fetch wrong instruction stream	r0p0, r1p0	r2p0
1130799	CatB	TLBI VAAE1 or TLBI VAAL1 targeting a page within hardware page aggregated address translation data in the L2 TLB might cause corruption of address translation data	r0p0, r1p0, r2p0	r3p0
1165347	CatB	Continuous failing STREX because of another core snooping from speculatively executed atomic behind constantly mispredicted branch might cause livelock	r0p0, r1p0, r2p0	r3p0
1165522	CatB	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation	r0p0, r1p0, r2p0	r3p0
1188873	CatB	MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result	r0p0, r1p0, r2p0	r3p0
1207823	CatB	The exclusive monitor might end up tracking an incorrect cache line in the presence of a VA-alias, causing a false pass on the exclusive access sequence	r0p0, r1p0, r2p0	r3p0
1220197	CatB	Streaming store under specific conditions might cause deadlock or data corruption	r0p0, r1p0, r2p0	r3p0
1257314	CatB	Multiple floating-point divides/square roots concurrently completing back-to-back and	r0p0, r1p0, r2p0, r3p0	r3p1

ID	Cat	Summary	Found in versions	Fixed in version
		flushing back-to-back might cause data corruption		
1262606	CatB	Concurrent instruction TLB miss and mispredicted branch instruction located at the end of 32MB region might fetch wrong instruction stream	r0p0, r1p0, r2p0, r3p0	r3p1
1262888	CatB	Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation	r0p0, r1p0, r2p0, r3p0	r3p1
1275112	CatB	A T32 instruction inside an IT block followed by a mispredicted speculative instruction stream might cause a deadlock	r0p0, r1p0, r2p0, r3p0	r3p1
1463225	CatB	Software Step might prevent interrupt recognition	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
1286807	CatB (rare)	Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-read ordering violation	r0p0, r1p0, r2p0, r3p0	r3p1
1418040	CatB (rare)	MRRC reads of some Generic Timer system registers in AArch32 mode might return corrupt data	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
901865	CatC	Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock	r0p0	r1p0
930203	CatC	Persistent error response to transactions issued on behalf of Page descriptor Access bit and Dirty bit updates might livelock	r0p0	r1p0
930239	CatC	Failure to report or incorrect reporting of L2 data RAM ECC errors	r0p0	r1p0
933092	CatC	Critical beat data for an L2 cache miss, poisoned or tagged with error, consumed by a load without reporting an abort	r0p0	r1p0
933779	CatC	DBGDTRTX register fails to hold value through Warm reset	r0p0	r1p0
934968	CatC	DCPSx instruction with SCTLR_EL1.IESB = 1 while in debug state might not execute correctly	r0p0	r1p0
944783	CatC	Address breakpoint might cause a deadlock with certain AArch32 T32 code sequences	r0p0	r1p0
973981	CatC	L2 might report multiple RAS errors for the same prefetch request	r0p0	r1p0
974001	CatC	Deferred errors might cause silent data corruption following a hardware update of Access and Dirty bits in a translation table entry	r0p0	r1p0

ID	Cat	Summary	Found in versions	Fixed in version
978245	CatC	Executing unallocated encoding in conversion between floating-point and integer instruction class does not generate Undefined Instruction exception	r0p0	r1p0
980456	CatC	Stuck-at-fault in L1 instruction cache data array might cause deadlock with certain AArch32 T32 code sequences	r0p0	r1p0
986709	CatC	MRS to DBGDTR_EL0 might cause EDSCR.RXfull bit to clear incorrectly	r0p0	r1p0
988575	CatC	Unaligned cache line split load to NC or Device memory, tagged with poison or external error on its first half, might cause data corruption	r0p0	r1p0
1051464	CatC	CTI trigger occurring on same cycle PREADYCD is received might cause CTI trigger to be missed	r0p0, r1p0	r2p0
1057923	CatC	Extra instruction might be executed during Halting Step when stepping WFI, WFE and some self-synchronizing system register writes	r0p0, r1p0	r2p0
1069401	CatC	Debug APB accesses to the ELA RAM might return incorrect data	r0p0, r1p0	r2p0
1085091	CatC	The ERXADDR_EL1 register might report an incorrect physical address for an L1 data tag RAM single-bit correctable ECC error	r0p0, r1p0, r2p0	r3p0
1096402	CatC	Exception packet for return stack match might return incorrect [E1:E0] field	r0p0, r1p0, r2p0	r3p0
1109624	CatC	Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock	r0p0, r1p0, r2p0	r3p0
1119735	CatC	16-bit T32 instruction close to breakpoint location might cause early breakpoint exception	r0p0, r1p0, r2p0	r3p0
1126105	CatC	Read from L1 instruction cache data array using RAMINDEX operation might return data from the wrong location	r0p0, r1p0, r2p0	r3p0
1144394	CatC	Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error	r0p0, r1p0, r2p0	r3p0
1145826	CatC	ERR0MISC0 might report incorrect BANK and SUBBANK values for parity errors in L1 instruction cache data array	r0p0, r1p0, r2p0	r3p0
1192279	CatC	IMPLEMENTATION DEFINED fault for unsupported atomic operations is not routed to proper Exception level	r0p0, r1p0, r2p0	r3p0

ID	Cat	Summary	Found in versions	Fixed in version
1214504	CatC	Direct access to L1 data TLB might report incorrect value of valid bit of the corresponding TLB entry	r0p0, r1p0, r2p0	r3p0
1220808	CatC	ERR0STATUS.SERR encoding is incorrect for error responses from slave and deferred data errors from slave which are not supported	r0p0, r1p0, r2p0	r3p0
1227053	CatC	Streaming writes to memory mapped Non-shareable and write-back might cause data corruption because of reordering	r0p0, r1p0, r2p0, r3p0	r3p1
1244984	CatC	Illegal return event might corrupt PSTATE.UA0	r0p0, r1p0, r2p0, r3p0	r3p1
1256788	CatC	Halting step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error	r0p0, r1p0, r2p0, r3p0	r3p1
1264383	CatC	Write-Back load after two Device-nG* stores to the same physical address might get invalid data	r0p0, r1p0, r2p0, r3p0	r3p1
1346756	CatC	TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0	r0p0, r1p0, r2p0, r3p0, r3p1, r4p0	Open
1349291	CatC	Uncontainable (UC) SError might be incorrectly logged as an Unrecoverable (UEU) SError	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
1356341	CatC	L1D_CACHE access related PMU events and L1D_TLB access related PMU events increment on instructions/micro-operations excluded from these events	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
1395332	CatC	Read from PMCCNTR in AArch32 might return corrupted data	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
1406411	CatC	MSR DSPSR_EL0 while in debug state might not correctly update PSTATE.{N,C,Z,V,GE} on debug exit	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
1408724	CatC	Portions of the branch target address recorded in ETM trace information might be incorrect for some branches immediately preceding an indirect branch with a malformed branch target address	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
1415323	CatC	Ordering violation might occur when a load encounters an L1 tag RAM single bit ECC error when a snoop request targets the same line	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
1430754	CatC	Write to External Debug Registers might cause a deadlock with certain AArch32 T32 code sequences	r0p0, r1p0, r2p0, r3p0, r3p1	r4p0
1487185	CatC	Waypoints from previous session might cause single-shot comparator match when trace enabled	r0p0, r1p0, r2p0, r3p0, r3p1, r4p0	Open

ID	Cat	Summary	Found in versions	Fixed in version
1490853	CatC	TRCIDR3.CCITMIN value is incorrect	r0p0, r1p0, r2p0, r3p0, r3p1, r4p0	Open

# Errata descriptions

## Category A

---

There are no errata in this category.

## Category A (rare)

---

### 1315703

#### Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-write ordering violation

##### Status

Fault Type: Programmer Category A Rare

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

##### Description

If a virtual address for a cacheable mapping of a location is being accessed by a core while another core is remapping the virtual address to a new physical page using the recommended break-before-make sequence, then under rare circumstances TLBI+DSB completes before a write using the translation being invalidated has been observed by other observers.

##### Configurations Affected

The erratum affects all multi-core configurations.

##### Conditions.

1. Core A has in program order a store (ST1) and a younger load (LD1) to the same cacheable virtual address.
2. Core B marks the associated translation table entry invalid, followed by a DSB; TLBI; DSB sequence which generates a sync request to Core A.
3. LD1 executes speculatively past ST1 and returns its result using the original physical address (PA1) under specific rare conditions before Core A has responded to the sync request.
4. At the time of receiving the sync request, on Core A:
  1. No load younger than ST1 has executed out-of-order for any of the following instructions:
    1. Load.
    2. DMB.
    3. DSB.
    4. Atomic instruction which updates a register and has acquire semantics.
  2. No store younger than ST1 has already computed its physical address (PA).
5. Any memory request from core A which was initiated prior to the sync request completes.
6. ST1 is not able to compute its PA before Core A responds to the sync request.
7. Core B receives the sync response and updates the translation table entry to map a new PA (PA2), which has write permissions and differs on bits [23:12] from PA1, followed by a DSB.
8. ST1 performs memory write using PA2 on Core A and commits the result from LD1 using PA1 because the read-after-write ordering violation between ST1 and LD1 is not detected.

##### Implications

If the above conditions are met under certain rare conditions, then this erratum might result in a read-after-write ordering violation.

##### Workaround

This erratum can be avoided by setting PSTATE.SSBS to 0 or CPUACTLR2\_EL1[16] to 1, hence preventing LD1 from speculating past ST1. This will have a performance impact on general workloads.

## Category B

---

### 905797

#### Failure to enforce read-after-read ordering rules

##### Status

Fault Type: Programmer Category B  
Fault Status: Present in r0p0. Fixed in r1p0.

##### Description

Under a combination of unusual conditions, it is possible for a younger load to bypass an older load to the same address, and for the two loads to observe updates to that address in the wrong order. In other words, the younger load might observe data which is globally ordered before the data that is observed by the older load.

##### Configurations Affected

This erratum affects all configurations.

##### Conditions

1. PE 1 is executing a program which contains a load atomic to physical address A, followed by two cacheable loads to physical address B. The load atomic does not have acquire semantics.
2. PE 2 is executing a program which modifies the value stored at physical address B.

If the above conditions are met under certain unusual timing conditions, then it is possible for the older of the two loads on PE 1 to observe the newly modified value from PE 2 while the younger load on PE 1 observes the value before the modification by PE 2.

##### Implications

If this erratum occurs, then multi-threaded software which relies on the read ordering rules might get an incorrect result.

##### Workaround

A workaround is not expected to be necessary in most cases. This is because the conditions require a combination of unusual timing conditions and load atomic instructions, which are not yet widely in use.

However, for systems using an ACE interconnect or a CHI interconnect that does not support far atomic operations (thus input **BROADCASTATOMIC** pin is tied to 0), setting CPUACTRL2\_EL1[2] to 1 will prevent the conditions necessary to hit this erratum.

## 961148

### Reads from DSU CLUSTER\* or ERX\* system registers might return corrupted data

#### Status

Fault Type: Programmer Category B  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

When a system register read from a particular set of system registers is executed speculatively, a subsequent read from the same set of registers might return corrupted data. The registers affected are the DynamIQ Shared Unit (DSU) CLUSTER\* control system registers, and the ERX\* error system registers when ERRSELR\_EL1.SEL=1.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. For ERX\* registers, the ERRSELR\_EL1.SEL is set to 1.
2. MRS Instruction A targeting a CLUSTER\* or ERX\* register is speculatively executed.
3. MRS Instruction B targeting a CLUSTER\* or ERX\* register is executed before the core receives the read response from Instruction A.

Note that the registers in the first MRS read and second MRS read do not have to be the same for this erratum to occur.

#### Implications

If the above conditions are met, data returned from the MRS instruction B targeting a CLUSTER\* or ERX\* register might be corrupted.

#### Workaround

In most cases, this erratum can be avoided by inserting an ISB instruction before the MRS to the CLUSTER\* or ERX\* system register, as the ISB can prevent the MRS from being speculatively executed.

## 977072

### Accessing certain Debug or Generic Timer system registers in AArch32 might cause incorrect system register values

#### Status

Fault Type: Programmer Category B  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

Conditional MRC/MCR/MRRC/MCRR accesses, or speculative unconditional MRC/MRRC reads, to certain Debug or Generic Timer system registers in AArch32 state can result in incorrect values for these system registers.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is executing in AArch32 state.
2. A non-exceptional MRC/MCR/MRRC/MCRR access is made to one of the following registers: DBGDTRTXint, DBGDTRRXint, CNTP\_TVAL, CNTP\_CTL, CNTV\_TVAL, CNTV\_CTL, CNTPCT, CNTVCT, CNTP\_CVAL, or CNTV\_CVAL.

#### Implications

If the above conditions are met, then a read of an affected register might return an incorrect result, and a write of an affected register might occur unconditionally.

#### Workaround

The erratum can be avoided by trapping MRC/MCR/MRRC/MCRR accesses in AArch32 to the affected registers and doing the equivalent code sequence in the trap handler. To trap the CNT\* accesses, set CNTKCTL\_EL1.{ELOPTEN, EL0VTEN, EL0VCTEN, EL0PCTEN} to 0. To trap the DBG\* accesses, set MDSCR\_EL1.TDCC to 1.

## 981980

### Interrupt is taken immediately after MSR DAIF instruction masks the interrupt

#### Status

Fault Type: Programmer Category B  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

When an interrupt arrives during the execution of an instruction that causes the interrupt to be masked, either MSR DAIFSet (immediate) or MSR DAIF (register), in some circumstances the interrupt will be erroneously taken on the instruction immediately following the MSR. Under the simple sequential execution model, that interrupt should not be taken because it has just been masked. In the interrupt handler, SPSR\_ELx and ELR\_ELx will reflect the fact that the relevant PSTATE.{A,I,F} mask bit was set when the interrupt was taken.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. An MSR DAIFSet (immediate) or MSR DAIF (register) instruction that changes the relevant PSTATE.{A,I,F} bit from 0 to 1 is executing.
2. An interrupt arrives during the execution of that MSR instruction and the execution state of the machine is such that the decision to take that interrupt depends on the PSTATE.{A,I,F} bit.
3. The interrupt is taken on the next instruction after the MSR instruction although it is newly-masked and should not be taken.

#### Implications

If the above conditions are met, then the interrupt is incorrectly taken on the instruction following the MSR. The SPSR\_ELx.{A,I,F} bits and ELR\_ELx will indicate that the masking MSR executed before taking the interrupt.

#### Workaround

Generally, it is expected that software will be robust against taking an interrupt immediately after masking it in PSTATE. If not, then a workaround is to subtract 4 from the ELR and clear the relevant mask bit in the SPSR when the interrupt vector is entered erroneously.

## 1043202

### AArch32 T32 CLREX in an IT block will clear exclusive monitor even if it fails condition code check

#### Status

Fault Type: Programmer Category B  
Fault Status: Present in r0p0, r1p0. Fixed in r2p0.

#### Description

The AArch32 T32 CLREX instruction in an IT block will always clear the exclusive monitor, even when the CLREX condition code fails.

Note: The CLREX instruction does not have a condition code outside of an IT block.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is executing in AArch32 state.
2. A T32 CLREX instruction is in an IT block, and the CLREX fails the condition code check while the exclusive monitor is in the exclusive state.

#### Implications

If the above conditions are met, then a subsequent store-exclusive might unexpectedly fail.

#### Workaround

This erratum can be avoided by replacing all T32 CLREX instructions with an ISB instruction. This can be done through the following write sequence to several IMPLEMENTATION DEFINED registers:

```
LDR x0,=0x0
MSR S3_6_c15_c8_0,x0 ; MSR CPUPSELR_EL3, X0
LDR x0,=0xF3BF8F2F
MSR S3_6_c15_c8_2,x0 ; MSR CPUPOR_EL3, X0
LDR x0,=0xFFFFFFFF
MSR S3_6_c15_c8_3,x0 ; MSR CPUPMR_EL3, X0
LDR x0,=0x800200071
MSR S3_6_c15_c8_1,x0 ; MSR CPUPCR_EL3, X0
ISB
```

## 1073348

### Concurrent instruction TLB miss and mispredicted return instruction might fetch wrong instruction stream

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0. Fixed in r2p0.

#### Description

When branches are speculatively executed, either the prediction is correct and the following speculative instruction stream is architecturally executed, or the branch is mispredicted and the pipeline is flushed. Under certain conditions, if an unconditional indirect branch is speculatively executed while a translation table walk response is almost complete, then the speculative instruction stream might not be flushed if the branch was incorrectly predicted.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. One of the following unconditional indirect branches is speculatively executed where the branch target crosses a 32MB boundary:
  - A64: RET.
  - A32: BX lr; POP {...,pc}; LDMIA r13!, {...,pc}; LDR PC, [SP], #offset.
  - T32: BX lr; POP {...,pc}; LDMIA r13!, {...,pc}; LDR PC, [SP], #offset.
2. A translation table walk response arrives around the time the branch is speculated.
3. When the branch resolves, bits[24:0] of the speculated address must match the actual address.

#### Implications

If the above conditions are met, then the core might execute the wrong instruction stream.

#### Workaround

This erratum can be avoided by setting CPUACTLR\_EL1[6] to 1, which disables static prediction.

## 1130799

### TLBI VAAE1 or TLBI VAALE1 targeting a page within hardware page aggregated address translation data in the L2 TLB might cause corruption of address translation data

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

TLBI VAAE1 or TLBI VAALE1 targeting a page within aggregated address translation data in the L2 TLB invalidates the page, but might also corrupt the translation for other pages in the group. A subsequent translation miss request to a different page within the aggregated group might result in incorrect translation.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Full hardware page aggregation is enabled by setting CPUACTLR\_EL1[48] to 1.
2. Multiple ASIDs have aggregated address translation data in different ways of the L2 TLB for the same VA range.
3. TLBI VAAE1 or TLBI VAALE1 targeting a page within the aggregated address translation data is executed.

#### Implications

If the above conditions are met, then the MMU might generate incorrect translation.

#### Workaround

This erratum can be avoided by setting CPUACTLR2\_EL1[59] to 1. Setting CPUACTLR2\_EL1[59] to 1 might have a small impact on performance.

## 1165347

### Continuous failing STREX because of another core snooping from speculatively executed atomic behind constantly mispredicted branch might cause livelock

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

Under certain conditions, a loop might continuously mispredict. If the speculative instruction path has an atomic instruction to the same physical address as another core's exclusive monitor address, then this might cause a repeatable loop where the cache line is requested by the atomic instruction to be unique, opening the exclusive monitor on the other core.

#### Configurations Affected

The erratum affects all configurations.

#### Conditions

1. There is a loop that has a branch that is consistently mispredicted.
2. There is an atomic instruction outside of the loop that has the same physical address as the exclusive monitor address of another core, within a cache line. The atomic instruction makes a unique request, snooping that cache line from other cores, and opening the exclusive monitor.

#### Implications

If the above conditions are met, the core might livelock.

#### Workaround

This erratum can be avoided by setting CPUACTLR2\_EL1[0] to 1 and CPUACTLR2\_EL1[15] to 1.

## 1165522

### Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

A speculative Address Translation (AT) instruction translates using registers associated with an out-of-context translation regime and caches the resulting translation in the L2 TLB. A subsequent translation request generated when the out-of-context translation regime is current uses the previous cached L2 TLB entry producing an incorrect virtual to physical mapping.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. A speculative AT instruction performs a table walk translating virtual address to physical address using registers associated with an out-of-context translation regime.
2. Address translation data generated during the walk is cached in the L2 TLB.
3. The out-of-context translation regime becomes current and a subsequent memory access is translated using previously cached address translation data in the L2 TLB, resulting in an incorrect virtual to physical mapping.

#### Implications

If the above conditions are met, the resulting translation would be incorrect.

#### Workaround

When context-switching the register state for an out-of-context translation regime, system software at EL2 or above must ensure that all intermediate states during the context-switch would report a level 0 translation fault in response to an AT instruction targeting the out-of-context translation regime. Note that a workaround is only required if the system software contains an AT instruction as part of an executable page.

## 1188873

### MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

Under certain internal timing conditions, an MRC instruction that closely follows an MRRC instruction might produce incorrect data when the MRRC is a read of specific Generic Timer system registers in AArch32 state.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is executing at AArch32 EL0.
2. An MRRC instruction which reads either the CNTPCT, CNTVCT, CNTP\_CVAL, or CNTV\_CVAL register is executed.
3. An MRC instruction is executed.

#### Implications

If this erratum occurs, then the destination register of the MRC is incorrect.

#### Workarounds

The erratum can be avoided by trapping MRC/MCR/MRRC/MCRR accesses in AArch32 to the affected registers and doing the equivalent code sequence in the trap handler. To trap the CNT\* accesses, set CNTKCTL\_EL1.{ELOPTEN, EL0VTEN, EL0VCTEN, ELOPCTEN} to 0. If HCR\_EL2.{E2H,TGE}={1,1} then set CNTHCTL\_EL2.{ELOPTEN, EL0VTEN, EL0VCTEN, ELOPCTEN} to 0. The following registers will be trapped: CNTP\_CTL, CNTP\_CVAL, CNTP\_TVAL, CNTV\_CTL, CNTV\_CVAL, CNTV\_TVAL, CNTPCT, CNTVCT, CNTFRQ.

## 1207823

### The exclusive monitor might end up tracking an incorrect cache line in the presence of a VA-alias, causing a false pass on the exclusive access sequence

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

Under certain conditions, the exclusive monitor that tracks the Physical Address (PA) for the exclusive-access sequence, might end up tracking the incorrect way the cache line is in the L1 cache. As a result, a subsequent STREX might get a false pass, even though the cache line was written to by another master.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. There is a load preceding the LDREX/STREX loop that has the same PA as the exclusive monitor address, within a cache line. However the load has a different VA, specifically a different VA[13:12] for 64KB L1 cache.
2. The LDREX issues ahead of this older load, misses the L1, and makes a request out to the L2 by allocating a request buffer. The L2 responds to the request for the LDREX, the line is allocated into the L1 cache, but the LDREX is prevented from picking up the response.
3. The older load subsequently misses the L1 and makes a request to the L2, using the same request buffer as that was previously used by the LDREX.
4. If the LDREX now replays, such that it coincides with the L2 response for the older load with the same PA, but a different VA, then it can forward from the L2 response for this load and complete. At this point, the exclusive monitor ends up capturing the way that this VA-aliased load is allocated into the L1, but the correct index that corresponds to the LDREX.
5. The exclusive monitor now ends up tracking the incorrect cache line. If the line was snooped out, it would therefore not transition to the open state.

#### Implications

If the above conditions are met, then the core might allow a subsequent STREX to pass, even though the LDREX/STREX sequence was not atomic.

#### Workaround

This erratum can be avoided by setting CPUACTLR2\_EL1[11] to 1.

## 1220197

### Streaming store under specific conditions might cause deadlock or data corruption

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

Under certain rare conditions, a streaming write of at least 64 consecutive bytes might send only 32 bytes of data from the L1 data cache to higher level caches.

#### Configurations Affected

The erratum affects all configurations.

#### Conditions

1. A store to address A is dispatched down a speculative path, before the write stream was engaged.
2. The write stream was engaged for a full cache line write.
3. A younger store instruction with address A is dispatched.

#### Implications

If the above conditions are met under certain timing conditions, then this erratum might result in deadlock or data corruption.

#### Workaround

This erratum can be avoided by setting CPUCTLR\_EL1[25:24] to 0b11, which disables write streaming to the L2. This will have an impact on performance for streaming workloads.

## 1257314

### Multiple floating-point divides/square roots concurrently completing back-to-back and flushing back-to-back might cause data corruption

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

#### Description

Under certain conditions, two floating-point divide or square root instructions completing back-to-back and concurrently getting flushed by back-to-back branch mispredicts might result in data corruption.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Two or more concurrently executing floating-point divide and/or square root instructions need to complete in back-to-back cycles.
2. A branch mispredict arrives concurrently with the completion of the first divide. This divide will flush.
3. Another branch mispredict arrives concurrently with the completion of the second divide. This divide will flush.
4. No other floating-point/vector instructions are in the scheduler to be issued.
5. Newly dispatched instructions coincidentally pick up a register resource that was freed up by the last flushed divide.
6. The newly dispatched instruction gets issued before its producer is issued.

#### Implications

If the above conditions are met, then this erratum might result in data corruption.

#### Workaround

This erratum can be avoided by setting CPUACTLR3\_EL1[10] to 1, which prevents parallel execution of divide and square root instructions.

## 1262606

### Concurrent instruction TLB miss and mispredicted branch instruction located at the end of 32MB region might fetch wrong instruction stream

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

#### Description

When branches are speculatively executed, either the prediction is correct and the following speculative instruction stream is architecturally executed, or the branch is mispredicted and the pipeline is flushed. Under certain conditions, if a branch located at the end of a 32MB region is speculatively executed while a translation table walk response is almost complete, then the speculative instruction stream might not be flushed if the branch was incorrectly predicted.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. A branch instruction located at the end of a 32MB region (if the branch address is  $PC=X[63:0]$ ,  $PC[24:6]=\text{all1}$ ) is speculatively executed.
2. The target of the speculatively executed branch belongs to neither the current 32MB region ( $PC[63:25] = X[63:25]+0$ ) nor the next sequential region ( $PC[63:25] = X[63:25]+1$ ).
3. A translation table walk response for the other 32MB region arrives around the time the branch is speculated and written into the L1 instruction TLB.
4. When the branch resolves, bits[24:0] of the speculated address must match the actual address.

#### Implications

If the above conditions are met, then the core might execute the wrong instruction stream.

#### Workaround

This erratum can be avoided by setting `CPUACTLR_EL1[13]` to 1, which delays instruction fetch after branch misprediction. This workaround will have a small impact on performance.

## 1262888

### Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

#### Description

Under specific conditions, an incorrect virtual to physical mapping might happen because the L2 TLB is corrupted. The L2 TLB might be corrupted because of both:

- A translation access hitting an entry in the L2 TLB which was previously allocated by the MMU hardware prefetch mechanism.
- A TLBI VAAE1 or TLBI VAALE1 for a non-active VMID context invalidating an entry.

A subsequent translation which hits against the corrupted entry generates an incorrect translation.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Virtualization is enabled.
2. The MMU hardware prefetcher installs an entry in the L2 TLB.
3. A translation request in the current VMID context hits on the prefetched entry.
4. TLBI VAAE1 or TLBI VAALE1 targeting a page within a non-active VMID context is in the process of invalidating a page.

#### Implications

If the above conditions are met, then the MMU might generate an incorrect translation.

#### Workaround

This erratum can be avoided by setting CPULECTLR\_EL1[51] to 1, which disables the MMU hardware prefetcher. Setting this bit might have a small impact on performance.

## 1275112

### A T32 instruction inside an IT block followed by a mispredicted speculative instruction stream might cause a deadlock

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

#### Description

The core might hang when it executes a T32 instruction inside an IT block.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. A T32 instruction is inside an IT block.
2. Subsequent instructions repeatedly create branch misprediction. Branch predictor misprediction occurs either because:
  1. Address translation is disabled.
  2. The second half of the T32 instruction can be decoded as 16-bit instruction updating R15 (PC).
  3. Branch predictor RAMs have soft errors.
3. Another IT block instruction is fetched from the speculative instruction stream (that is corrected by the above branch misprediction) and executed before the first T32 instruction is retired from pipeline.

#### Implications

If the above conditions are met, the core might deadlock as the instruction in the IT block does not complete.

#### Workaround

This erratum can be avoided by setting CPUACTLR\_EL1[13] to 1, which delays instruction fetch after branch misprediction. This workaround will have a small impact on performance.

The workaround for this erratum is the same as the workaround for erratum 1262606.

## 1463225

### Software Step might prevent interrupt recognition

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

The Software Stepping of a system call instruction (SVC, HVC, or SMC) can prevent recognition of subsequent interrupts when Software Stepping is disabled in the exception handler of the system call. Additionally, unconventional code involving the Software Stepping of an MSR instruction that clears the MDSCR\_EL1.SS bit (disables Software Step while stepping) can prevent recognition of subsequent interrupts.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions:

##### Case A:

1. Software Step is enabled.
2. The system configuration is (MDSCR\_EL1.KDE==1) or (MDSCR\_EL1.KDE==0 and HCR\_EL2.E2H==1 and (HCR\_EL2.TGE==1 or MDSCR\_EL2.TDE==1)).
3. An ERET with SPSR\_ELx.SS==1 is executed to cause the Software Step state machine to enter the active-not-pending state.
4. A system call instruction (SVC, HVC, or SMC) is executed and generates its system call exception (that is, it is not trapped).
5. The exception handler of the system call disables Software Step by clearing MDSCR\_EL1.SS or by setting SPSR\_ELx.D such that, upon return, no Software Step exception is taken.

##### Case B:

1. Software Step is enabled.
2. An ERET with SPSR\_ELx.SS==1 is executed to cause the Software Step state machine to enter the active-not-pending state.
3. An MSR MDSCR\_EL1 instruction that clears the MDSCR\_EL1.SS bit is executed (disables Software Step).

#### Implications

##### Case A:

Arm believes that for this product, MDSCR\_EL1.KDE is not set to 1 by deployed devices in the field and is only used when debugging the system software during initial product development. In these cases, the effect of the erratum is for interrupts to be disabled even after switching to other software contexts that are not being debugged as part of the system software debugging. Arm believes that a workaround does not need to be deployed for the situation where MDSCR\_EL1.KDE==1, and a workaround is not available.

Some devices are expected to run an operating system at EL2 with HCR\_EL2.E2H set to 1. The implication of this erratum for such a system is that single-stepping of an untrusted user application at EL0 can lead to subsequent execution not recognizing interrupts where it should, leading to errant behavior. The software workaround described below can be deployed in the operating system at EL2 to prevent single-stepping of untrusted user applications from triggering this erratum.

##### Case B:

Unconventional code involving the Software Stepping of the disabling instruction is not expected to be encountered, therefore no workaround is required.

**Workaround**

When Software Step is used to debug an application under an operating system running at EL2 with HCR\_EL2.E2H set to 1, the software workaround involves explicitly triggering a Software Step exception with modifications to the system call exception handler code and Software Step exception handler code. This entails setting MDSCR\_EL1.KDE and MDSCR\_EL1.SS and clearing PSTATE.D to trigger a Software Step exception from the system call handler. The Software Step handler then sets SPSR\_ELx.D before returning back to the system call handler, where MDSCR\_EL1.KDE and MDSCR\_EL1.SS are restored to their original values.

If a workaround is required when MDSCR\_EL1.KDE is set to 1, then please contact Arm.

## Category B (rare)

---

**1286807**

**Modification of the translation table for a virtual page which is being accessed by an active process might lead to read-after-read ordering violation**

**Status**

Fault Type: Programmer Category B Rare

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

**Description**

If a virtual address for a cacheable mapping of a location is being accessed by a core while another core is remapping the virtual address to a new physical page using the recommended break-before-make sequence, then under very rare circumstances TLBI+DSB completes before a read using the translation being invalidated has been observed by other observers.

**Configurations Affected**

The erratum affects all multi-core configurations.

**Conditions**

1. Core A speculatively executes a load (LD2) ahead of an older load (LD1) to the same cacheable virtual address.
2. Core B marks the associated translation table entry invalid, followed by a DSB; TLBI; DSB sequence which generates a sync request.
3. LD2 returns its result using the original physical address (PA1) under specific narrow timing conditions before Core A has responded to the sync request.
4. Core B receives the response and updates the translation table entry to map a new physical address (PA2) followed by a DSB.
5. LD1 returns its result using PA2 on Core A and commits the result from LD2 using PA1 because the read-ordering violation is not detected.

**Implications**

If the above conditions are met under certain timing conditions, then this erratum might result in a read ordering violation.

**Workaround**

This erratum can be avoided by executing the TLB invalidate and DSB instructions a second time before modifying the translation table of a virtual page that is being accessed by an active process.

## 1418040

### MRRC reads of some Generic Timer system registers in AArch32 mode might return corrupt data

#### Status

Fault Type: Programmer Category B Rare

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

An MRRC read of certain Generic Timer system registers in AArch32 mode might return corrupt data.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

This erratum occurs when the following conditions are met under rare internal timing conditions:

1. The core is executing at AArch32 at EL0.
2. An MRRC to CNTPCT, CNTVCT, CNTP\_CVAL, or CNTV\_CVAL is executed.

#### Implications

If the erratum occurs, then the second destination register [Rt2] of the MRRC will incorrectly contain the same data as the first destination register [Rt].

#### Workarounds

The erratum can be avoided by trapping MRC/MCR/MRRC/MCRR accesses in AArch32 to the affected registers and doing the equivalent code sequence in the trap handler.

To trap the CNT\* accesses, set CNTKCTL\_EL1.{EL0PTEN, EL0VTEN, EL0VCTEN, EL0PCTEN} to 0. If HCR\_EL2.{E2H,TGE}={1,1} then set CNTHCTL\_EL2.{EL0PTEN, EL0VTEN, EL0VCTEN, EL0PCTEN} to 0.

The following registers will be trapped:

- CNTP\_CTL.
- CNTP\_CVAL.
- CNTP\_TVAL.
- CNTV\_CTL.
- CNTV\_CVAL.
- CNTV\_TVAL.
- CNTPCT.
- CNTVCT.
- CNTFRQ.

## Category C

---

### 901865

#### Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r1p0.

**Description**

Under certain conditions, an LDREX/STREX loop might continuously mispredict. If the speculative instruction path has a load or store to the same Physical Address (PA) as the exclusive monitor address, but with a different Virtual Address (VA), this might cause a repeatable loop where the cache line is lost, opening the exclusive monitor.

**Configurations Affected**

The erratum affects all configurations.

**Conditions**

1. The LDREX/STREX loop has a branch that is consistently mispredicted. This includes all Device memory code, which the branch predictor does not train.
2. There is a load or store outside of the loop that has the same PA as the exclusive monitor address, within a cache line. However, this load or store has a different VA, specifically VA[13:12] for 64KB L1 cache. The load or store makes a request to the L2 that snoops the L1, opening the exclusive monitor. The Arm architecture disallows a load or store inside an LDREX/STREX loop with VA aliasing to the exclusive monitor cache line.

**Implications**

If the above conditions are met, the core might livelock.

**Workaround**

This erratum is not expected to be encountered in real software. There is no workaround for this erratum.

## 930203

### Persistent error response to transactions issued on behalf of Page descriptor Access bit and Dirty bit updates might livelock

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

A Page descriptor Access bit and Dirty bit update request by the MMU might not be successful if accessing the memory location encounters an uncorrectable error. In this case, the MMU might retry the request repeatedly instead of reporting an external abort.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core cache has the descriptor in SharedClean state.
2. A ReadUnique request is sent to the system to update the Access Flag bit or the Dirty bit of the descriptor.
3. And either of the following:
  - The system responds with a Non-Data Error (NDErr).
  - The core is configured with CORE\_CACHE\_PROTECTION set to FALSE and receives a Poisoned or DataError(DErr) response.

#### Implications

If the above conditions occur in a persistent manner, then the processor might livelock.

#### Workaround

For Non-Data Error conditions, there is no workaround, as these represent a permanent error. For Data Error and Poisoned responses, this erratum can be avoided by setting CPUACTLR2\_EL1[41] to 1. This forces the core to perform Access Flag bit and Dirty bit updates as far atomics, which causes an invalidation of the SharedClean copy in the core cache.

Note this is effective only when the system can support far atomics.

## 930239

### Failure to report or incorrect reporting of L2 data RAM ECC errors

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

For certain operation types, a data read from the L2 data RAMs caused by a cache line victimization might fail to report and log a RAS error if such data contains a single or double bit ECC error. In some cases, an error is reported, but the physical address recorded is incorrect.

#### Configurations Affected

This erratum affects all configurations with `CORE_CACHE_PROTECTION` set to `TRUE`.

#### Conditions

When performing a refill of the L2 cache on behalf of an instruction fetch, load, store, or table walk, a data read from the L2 data RAMs for the cache line being replaced encounters a single or double bit ECC error.

#### Implications

If this erratum occurs, either:

- The L2 data RAM ECC error is detected, but the error is not reported or logged in the CPU RAS registers.
- The error is reported and logged, but the wrong physical address is recorded.

Error recovery software will not be able to correctly determine the source of a data error.

Note that any required error propagation to consumers of the data from the L2 data RAMs in the form of poison occurs correctly.

#### Workaround

A partial workaround is possible. Setting `CPUACTLR2_EL1[45]` to 1 forces reporting and logging of all L2 data RAM ECC errors. However, the reported physical address might still be incorrect. Other recorded information, such as array, subarray, and index are correct. Setting `CPUACTLR2_EL1[45]` to 1 might have a small impact on performance.

## 933092

### Critical beat data for an L2 cache miss, poisoned or tagged with error, consumed by a load without reporting an abort

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

Under certain conditions, an external error or poison flagged on a critical beat response for a linefill might get dropped. As a result, a load that forwards data from that critical beat, might return data without signaling an abort.

#### Configurations Affected

The erratum affects all configurations.

#### Conditions

1. The Processing Element (PE) executes one or more load instructions that miss in both the L1 and L2 caches, and initiates a linefill request.
2. The system returns the critical beat tagged with either poisoned data or an external error indication.
3. The critical beat is returned to the Load/Store (LS) unit of the PE, and no more data beats are returned to the LS for three or more cycles.
4. Load instructions that can complete by getting their data from the critical beat, are sent down the pipeline so that they pick up data three or more cycles after it was returned to the LS.

#### Implications

If the above conditions are met, the PE might consume poisoned data or data tagged with an error, without signaling an abort.

#### Workaround

A workaround is not expected to be necessary in most cases, as this erratum will only cause a negligible increase in the failure in time (FIT) rate.

If a workaround is required, set CPUACTLR2\_EL1[43] to 1. This prevents critical beat forwarding, and ensures that the load will abort in case the system reports an error. Setting CPUACTLR2\_EL1[43] to 1 might have a small impact on performance.

**933779****DBGDTRTX register fails to hold value through Warm reset****Status**

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

**Description**

The DBGDTRTX register is architecturally required to hold value through a Warm reset. Because of this erratum, a reset connection error has it resetting the value on a Warm reset instead.

**Configurations Affected**

This erratum affects all configurations.

**Conditions**

DBGDTRTX is holding a value other than the initial reset value and a Warm reset occurs.

**Implications**

If this erratum occurs, the content of the DBGDTRTX register is inaccurate.

**Workaround**

There is no workaround for this erratum.

**934968****DCPSx instruction with SCTLR\_EL1.IESB = 1 while in debug state might not execute correctly****Status**

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

**Description**

A DCPS1, DCPS2, or DCPS3 instruction that is executed in debug state while SCTLR\_EL1.IESB is set to 1 might result in incorrect execution of the instruction.

**Configurations Affected**

This erratum affects all configurations.

**Conditions**

1. SCTLR\_EL1.IESB is set to 1.
2. The core is in debug state.
3. The core executes a DCPS1, DCPS2, or DCPS3 instruction.

**Implications**

If the above conditions are met, then this erratum might result in deadlock, data corruption, or produce other undesirable effects. However, this erratum will not result in violation of access controls, for example, this erratum will not result in the core making accesses to Secure memory from Non-secure mode.

**Workaround**

The erratum can be avoided by clearing SCTLR\_EL1.IESB before executing a DCPSx instruction in debug state. If the core is in a state where SCTLR\_EL1 writes are trapped, then up to three write attempts might be required where each attempt might be trapped to a higher Exception level.

## 944783

### Address breakpoint might cause a deadlock with certain AArch32 T32 code sequences

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

If an Address breakpoint is set on a T32 instruction, then under certain conditions the core might stop executing a few instructions before the Breakpoint exception should occur.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is executing in AArch32 T32 instruction state.
2. The breakpoint is set on a Cacheable line.
3. The breakpoint is not quadword aligned.
4. The cache line contains at least two 32-bit instructions, of which at least one must be after the breakpoint.

#### Implications

If the above conditions are met, the processor might deadlock.

#### Workaround

Any interrupt will break the processor out of the deadlock state. The deadlock can be avoided by forcing the page containing the T32 instruction to be Non-cacheable.

## 973981

### L2 might report multiple RAS errors for the same prefetch request

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

If a hardware or software prefetch targeting the L2 encounters a tag ECC error and hazards against an outstanding request for the same cache line, then multiple RAS errors might be reported.

#### Configurations Affected

This erratum affects all configurations with `CORE_CACHE_PROTECTION` set to `TRUE`.

#### Conditions

A hardware or software prefetch operation that hazards against an outstanding read request for the same cache line and detects a tag ECC error might allow the internal signals for RAS errors to remain asserted, leading to multiple reported errors for the same operation.

#### Implications

If this erratum occurs, then the `ERR0STATUS.OF` bit might be set when only one error has actually occurred.

#### Workaround

No workaround is required for this erratum.

## 974001

### Deferred errors might cause silent data corruption following a hardware update of Access and Dirty bits in a translation table entry

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

An L2 linefill caused by a hardware update to the Access and Dirty bits in the translation tables might cause silent data corruption if the data received from the system contains an external error or poison indication.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The configuration option CORE\_CACHE\_PROTECTION is set to FALSE or error correction is disabled with ERROCTLR.ED set to 0.
2. A memory access causes a hardware update of the Access and/or Dirty flags of the corresponding translation table entry.
3. The hardware update causes the L2 cache to initiate a linefill operation because of a cache miss or hit to SharedClean state.
4. The data returned from the system contains a deferred error indicated by poison, a data error, or non-data error responses in data other than the doubleword containing the Access and Dirty flags.

#### Implications

If the above conditions are met, the translation table Access and Dirty bits might not be updated if the cache line containing the translation table entry contains a deferrable data error.

#### Workaround

If the system supports far atomic accesses to cacheable memory, then setting CPUACTLR2\_EL1[41] to 1 forces the L2 cache to perform hardware updates of the Access and Dirty flags as far atomics.

## 978245

### Executing unallocated encoding in conversion between floating-point and integer instruction class does not generate Undefined Instruction exception

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

If the core executes the following unallocated encoding in the A64 conversion between floating-point and integer instruction class, where  $sf = x$ ,  $S = 0$ ,  $type = 11$ ,  $rmode = 01$ ,  $opcode = 11x$ , then instead of taking an Undefined Instruction exception, the core incorrectly executes this unallocated encoding as a vector half-precision "FABD <Vd>.<T>, <Vn>.<T>, <Vm>.<T>" instruction.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is executing in AArch64 state.
2. An unallocated encoding in the conversion between floating-point and integer instruction class, where  $sf = x$ ,  $S = 0$ ,  $type = 11$ ,  $rmode = 01$ ,  $opcode = 11x$ , is executed.

#### Implications

If the above conditions are met, the core does not take an Undefined Instruction exception.

#### Workaround

There is no workaround for this erratum.

## 980456

### Stuck-at-fault in L1 instruction cache data array might cause deadlock with certain AArch32 T32 code sequences

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

Detected parity errors in the L1 instruction cache data array will trigger a line fill request to repeat the instructions. In certain scenarios, the returned data is not used directly but is first stored in the cache. If a stuck-at-fault is present, then the core will continuously request the same line fill and no further instructions will be executed.

#### Configurations Affected

This erratum affects all configurations with `CORE_CACHE_PROTECTION` set to `TRUE`.

#### Conditions

1. The core is executing in AArch32 T32 instruction state.
2. The cache line contains a 32-bit instruction starting at odd halfword alignment.
3. The upper 5 bits of the second halfword of this 32-bit instruction must be `0b11101`, `0b11110`, or `0b11111`.
4. A stuck-at-fault must exist in the second halfword of the instruction.

#### Implications

If the above conditions are met, the processor might deadlock.

#### Workaround

Any interrupt will break the processor out of the deadlock state. The stuck-at-fault can be bypassed by forcing the page containing the T32 instruction to be Non-cacheable.

## 986709

### MRS to DBGDTR\_EL0 might cause EDSCR.RXfull bit to clear incorrectly

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

An MRS to DBGDTR\_EL0 which is speculatively executed might cause the EDSCR.RXfull bit to incorrectly clear before the data is read from DBGDTR\_EL0.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The EDSCR.RXfull bit is set to 1.
2. An MRS to DBGDTR\_EL0 is speculatively executed.

#### Implications

If the above conditions are met, then the EDSCR.RXfull bit might be cleared before the data is read from DBGDTR\_EL0. This might cause:

- The core to not receive all data when an external debugger sees the RXfull bit cleared and, as a result, sends new data before the core receives the old data.
- Earlier Instructions in the instruction stream to see the RXfull bit cleared out of program order.

#### Workaround

This erratum can be avoided by inserting an ISB instruction before the MRS to DBGDTR\_EL0, because the ISB can prevent the MRS from being speculatively executed.

## 988575

### Unaligned cache line split load to NC or Device memory, tagged with poison or external error on its first half, might cause data corruption

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

Under certain conditions, an external error or poison flagged on the first half of an unaligned load to Non-Cacheable (NC) or Device memory that crosses a cache line boundary, might get dropped. As a result, the unaligned NC or Device load can return data without signaling an abort.

#### Configurations Affected

The erratum affects all configurations.

#### Conditions

1. The Processing Element (PE) executes an unaligned load to NC or Device memory that crosses a cache line boundary.
2. The system returns data for the first line tagged with either poisoned data or an external error indication.
3. The second half of the cache line split is not tagged with either poisoned data or an external error.
4. The load, on receiving data for the first half, executes such that the second half of the unaligned load gets squashed by either an older load or some other high priority requestor and will be replayed.

#### Implications

If the above conditions are met, the PE might consume poisoned data or data tagged with an error, without signaling an abort.

#### Workaround

There is no workaround for this erratum.

## 1051464

### CTI trigger occurring on same cycle **PREADYCD** is received might cause CTI trigger to be missed

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0. Fixed in r2p0.

#### Description

A CTI trigger arriving on the same cycle the core debug block receives the **PREADYCD** from an outstanding CTI trigger transaction might result in the arriving CTI trigger to be lost. This erratum occurs only on CTI trigger events from the core to the external debug block.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Multiple CTI triggers are generated in the core going to the external debug block in close time proximity.
2. A CTI trigger event arrives in the same cycle the core debug block receives **PREADYCD** from an outstanding CTI trigger transaction.

#### Implications

If the above conditions are met, then the arriving CTI trigger might be lost and the system does not receive a trigger event. If the trigger event is from the same CTI source, then no issues will be seen as CTI triggers are allowed to be merged. If the trigger is from a different CTI source, then the debug system might not behave in an optimal fashion although debug operations can continue.

#### Workaround

No workaround is required for this erratum.

## 1057923

### Extra instruction might be executed during Halting Step when stepping WFI, WFE, and some self-synchronizing system register writes

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0. Fixed in r2p0.

#### Description

During Halting Step, execution of a small set of instructions in the Active-not-pending state can result in the execution of that instruction and the next instruction before returning control to the debugger by entering debug state. That is, instead of a single instruction executed between returns to the debugger, two instructions are executed. The set of instructions that can cause the stepping of an extra instruction is WFE, WFI, and some self-synchronizing system register writes.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Core is in Halting Step mode.
2. The instruction being stepped is either a WFE, a WFI, or some self-synchronizing system register writes.

#### Implications

If the above conditions are met, then two instructions will be stepped when a single step is expected, causing a potential DLR\_EL0 mismatch by software. However, the instructions still execute in the correct order and function correctly.

#### Workaround

There is no workaround for this erratum.

## 1069401

### Debug APB accesses to the ELA RAM might return incorrect data

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0. Fixed in r2p0.

#### Description

Debug APB registers support direct write and read accesses to the ELA RAM, using the following registers:

- RAM Write Address Register (RWAR).
- RAM Write Data Register (RWDR).
- RAM Read Address Register (RRAR).
- RAM Read Data Register (RRDR).

After writing the ELA RAM using the RWAR/RWDR registers, a subsequent read access using the RRAR/RRDR registers might return old data from the RRDR register. The ELA RAM read operation is dropped and the previous contents of the RRDR register are returned instead of the contents associated with the current operation.

#### Configurations Affected

This erratum affects all configurations with ELA set to TRUE.

#### Conditions

1. Write RWAR register with target index of ELA RAM.
2. Write RWDR register with write data to trigger the ELA RAM write access.
3. Write RRAR register with target index of ELA RAM to trigger the ELA RAM read access.
4. Read RRDR register to return the data.

#### Implications

If the above conditions are met, then old data is returned from the RRDR register because the write to the RRAR register to trigger the ELA RAM read access is dropped.

#### Workaround

This erratum can be avoided by inserting two writes to the ELA Lock Access Register (LAR) before writing the RRAR register.

## 1085091

### The ERXADDR\_EL1 register might report an incorrect physical address for an L1 data tag RAM single-bit correctable ECC error

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

If a load, store, prefetch, or snoop request encounters a single-bit correctable ECC error in the L1 data cache tag RAM, then the physical address captured in the ERXADDR\_EL1 register might be incorrect, even if the ERXSTATUS\_EL1.AV bit is set to 1, indicating a valid address.

#### Configurations Affected

This erratum affects all configurations with CORE\_CACHE\_PROTECTION set to TRUE.

#### Conditions

1. An L1 data cache tag RAM lookup on behalf of a load, store, prefetch, or snoop request encounters a single-bit correctable ECC error.
2. The address of the line that has the ECC error is not the address that is being looked up in the L1 data cache tag RAM.

#### Implications

If this erratum occurs, then the ERXSTATUS\_EL1.AV bit is set to 1, but the address captured in the ERXADDR\_EL1 register is not the correct physical address of the line that had the single-bit correctable ECC error.

#### Workaround

There is no workaround for this erratum.

## 1096402

### Exception packet for return stack match might return incorrect [E1:E0] field

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

When an abort or trap is taken at the target of an indirect branch matching the return stack value in the core ETM, an Exception packet might be generated with the 2-bit field [E1:E0] = 0b10, which implies an Address element before the Exception element. When there is a trace return stack match, an Address element should not be generated before the Exception element. With [E1:E0] = 0b10, the external Trace Analyzer might read the trace packet sequence to expect an Address element output before the Exception element and not complete the stack pop, which is incorrect. The correct value in the [E1:E0] field in the Exception packet for this case, should be 0b01.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. ETM is enabled.
2. TRCCONFIG.RS = 1, which indicates the return stack is enabled.
3. Abort or trap is taken at the target of an indirect branch matching the return stack.

#### Implications

If the above conditions are met, then the external Trace Analyzer does not pop on the return stack match, causing it to go out of sync with the core ETM.

#### Workaround

If tracing only EL0, then no workaround is required.

Otherwise, setting TRCCONFIG.RS = 0 to disable return stack is the workaround.

## 1109624

### Continuous failing STREX with VA alias access outside mispredicted exclusive sequence (LDREX/STREX) loop might cause livelock

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

Under certain conditions, an LDREX/STREX loop might continuously mispredict. If the speculative instruction path has a load or store to the same Physical Address (PA) as the exclusive monitor address, but a different Virtual Address (VA), then this might cause a repeatable loop where the cache line is lost, opening the exclusive monitor.

#### Configurations Affected

The erratum affects all configurations.

#### Conditions

1. The LDREX/STREX loop has a branch that is consistently mispredicted. This includes all Device memory code, which the branch predictor does not train.
2. There is a load or store outside of the loop that has the same PA as the exclusive monitor address, within a cache line. However, this load or store has a different VA, specifically VA[13:12] for 64KB L1 cache. The load or store makes a request to the L2 that snoops the L1, opening the exclusive monitor. The Arm architecture disallows a load or store inside an LDREX/STREX loop with VA aliasing to the exclusive monitor cache line.
3. The LDREX-STREX loop also contains an ISB instruction.

#### Implications

If the above conditions are met, then the core might livelock.

#### Workaround

This erratum is not expected to be encountered in real software. There is no workaround for this erratum.

## 1119735

### 16-bit T32 instruction close to breakpoint location might cause early breakpoint exception

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

If an address breakpoint is set on the instruction following a 16-bit T32 instruction, then under certain conditions the core might trigger the breakpoint on that 16-bit T32 instruction. This can happen if there is a parity error on the 16-bit T32 instruction before the breakpoint, or if the 16-bit T32 instruction has different cacheability than prior instructions.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is executing an AArch32 T32 code sequence.
2. A breakpoint is set on the instruction following a 16-bit T32 instruction.
3. One of the following conditions is true:
  - The breakpoint instruction follows a 16-bit T32 instruction containing a parity error.
  - The breakpoint instruction and the prior 16-bit T32 instruction both belong to a cache line that has different cacheability than the previous cache line.

#### Implications

If the above conditions are met, then the breakpoint might be triggered on the preceding 16-bit T32 instruction.

#### Workaround

There is no workaround for this erratum. This situation can be detected by reading the contents of the appropriate ELR\_ELx register after the breakpoint exception has been taken.

## 1126105

### Read from L1 instruction cache data array using RAMINDEX operation might return data from the wrong location

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

The RAMINDEX operation can be used to read the L1 instruction cache data array contents using the Index and Way fields. Because of this erratum, bits of the Index field of the RAMINDEX operation are swapped and Index {Index[13:6],Index[4:3],Index[5]} is used instead of Index[13:3].

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

A RAMINDEX operation is performed targeting the L1 instruction cache data array.

#### Implications

Data read from the RAMINDEX operation targeting the L1 instruction cache data array might not come from the specified Index field of the RAMINDEX operation.

#### Workaround

The Index field for the RAMINDEX operation can be adjusted appropriately to access the desired L1 instruction cache data array entry.

## 1144394

### Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

During software step, execution of some load instructions in the Active-not-pending state might result in the execution of that instruction and the next instruction before returning control to debugger software by taking a software step exception, instead of returning after a single instruction executed.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is in software step mode.
2. The instruction being stepped is a load instruction that loads two or more destination registers.
3. Snoop invalidation of a cache line referenced by the load occurs during its execution, or an ECC error response occurs on the load.

#### Implications

If the above conditions are met, then two instructions can be stepped when a single step is expected, causing a potential ELR\_ELx mismatch by software. However, the instructions still execute in the correct order and function correctly.

#### Workaround

There is no workaround for this erratum.

**1145826****ERR0MISC0 might report incorrect BANK and SUBBANK values for parity errors in L1 instruction cache data array****Status**

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

**Description**

If a parity error is detected in the L1 instruction cache data array, then the error location might not be computed correctly. This results in incorrect BANK and SUBBANK information in the ERR0MISC0 register.

**Configurations Affected**

This erratum affects all configurations with CORE\_CACHE\_PROTECTION set to TRUE.

**Conditions**

A parity error is detected in the L1 instruction cache data array.

**Implications**

If the above conditions are met, then the BANK and SUBBANK fields of the ERR0MISC0 register might have incorrect information. This does not impact other fields in the ERR0MISC0 register that apply to the L1 instruction cache.

**Workaround**

There is no workaround for this erratum.

## 1192279

### **IMPLEMENTATION DEFINED fault for unsupported atomic operations is not routed to proper Exception level**

#### **Status**

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### **Description**

If the interconnect does not support atomic memory operations, then instructions which try to perform these to Non-cacheable or Device memory take an IMPLEMENTATION DEFINED fault with Data Fault Status Code of ESR\_ELx.DFSC = 0b110101. If the PE is executing at EL0 or EL1, Stage 2 translation is enabled, and HCR\_EL2.CD forces the final memory type to be Non-Cacheable, then this fault is not routed to EL2.

#### **Configurations Affected**

The erratum affects all configurations.

#### **Conditions**

1. The interconnect does not support atomic operations.
2. The PE is executing at EL0 or EL1.
3. There is an atomic instruction to memory which is mapped as Non-cacheable because Stage 2 translation is enabled and HCR\_EL2.CD is set.

#### **Implications**

If the above conditions are met, then the IMPLEMENTATION DEFINED fault with Data Fault Status Code of ESR\_ELx.DFSC = 0b110101 is not routed to EL2.

#### **Workaround**

There is no workaround for this erratum.

## 1214504

### Direct access to L1 data TLB might report incorrect value of valid bit of the corresponding TLB entry

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### Description

An IMPLEMENTATION DEFINED instruction that reads the contents of the L1 data TLB after a context switch might report an incorrect value of the valid bit for the corresponding TLB entry.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. An instruction to perform a direct access to the L1 data TLB is present in program order before a context switch event.
2. The read of the L1 data TLB contents as part of the direct access instruction occurs after the context switch.

#### Implications

If the above conditions are met, then an incorrect value might be reported for the valid bit of the L1 data TLB entry being accessed directly.

#### Workaround

This erratum can be avoided by inserting a DSB after every instruction that accesses the L1 data TLB directly.

## 1220808

### **ERR0STATUS.SERR encoding is incorrect for error responses from slave and deferred data errors from slave which are not supported**

#### **Status**

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r2p0. Fixed in r3p0.

#### **Description**

The ERR0STATUS.SERR field is updated incorrectly for Error responses from slave and Deferred errors from slave not supported at master. Error responses from the interconnect for copyback transactions should record ERR0STATUS.SERR = 0x12. Because of this erratum, they incorrectly record 0x18. Undeferable data errors received from the interconnect should record ERR0STATUS.SERR = 0x15. Because of this erratum, they incorrectly record 0x12.

#### **Configurations Affected**

This erratum affects all configurations.

#### **Conditions**

This erratum occurs if one of the following conditions is true:

- The core issues a copyback transaction (WriteBackFull, WriteEvictFull, Evict, or WriteNoSnpFull) which then receives an error response.
- The core receives data containing an error (Poison or DErr response), but the core caches cannot defer the error by marking the data as poisoned in its caches. This occurs when the core is configured with CORE\_CACHE\_PROTECTION set to FALSE, or when ERR0CTLR.ED is 0.

#### **Implications**

If either of the above conditions are met, then the ERR0STATUS.SERR field is incorrect and software handling these errors reports the wrong class of error.

#### **Workaround**

There is no workaround for this erratum.

## 1227053

### Streaming writes to memory mapped Non-shareable and write-back might cause data corruption because of reordering

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

#### Description

Writes to contiguous bytes might be coalesced into one streaming write of 64 bytes. If such writes are performed to memory mapped Non-shareable and write-back, then two streaming writes to the same physical address might be performed in the wrong order.

#### Configurations Affected

This erratum affects configurations without a DSU L3 cache and snoop filter. Such systems are defined as direct connect using the following RTL parameter values:

- L3\_CACHE: FALSE.
- ACE: FALSE.
- PORTER\_SAM: TRUE.
- ACP: FALSE.
- PERIPH\_PORT: FALSE.
- ASYNC\_BRIDGE: TRUE.

#### Conditions

Write stream operations to memory mapped Non-shareable and write-back, or shareable and write-back with the \*BROADCASTOUTER \*pin deasserted can allocate the L2 cache without issuing a request on the CHI interface. This creates the possibility of two concurrent pending WriteNoSnpFull transactions of the same cache line on CHI without the proper sequencing to guarantee their order of performance.

#### Implications

If the above conditions are met, then the coalesced writes might be performed in the wrong order as determined by the sequential execution model.

#### Workaround

This erratum can be avoided by mapping all write-back memory as Inner or Outer Shareable.

## 1244984

### Illegal return event might corrupt PSTATE.UAO

#### Status

Fault Type: Programmer Category C

Fault Status: Present on r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

#### Description

An illegal return event from AArch64 state erroneously updates PSTATE.UAO from the saved process state bit[23] when the saved process state stipulates an intended return to AArch32. The correct behavior is to leave PSTATE.UAO unchanged.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

- An illegal return event from AArch64 state occurs. This involves at least one of the following, where the saved process state stipulates return to a mode or state that is illegal:
  - Execution of an ERET instruction.
  - Execution of a DRPS instruction in Debug state.
  - Exit from Debug state.
- The saved process state specifies the AArch32 target execution state. The saved process state bit, M[4], is 1.

#### Implications

PSTATE.UAO might be corrupted.

This corrupted value is saved in SPSR\_ELx on taking an Illegal Execution state exception or an asynchronous exception immediately after the illegal return event. The corrupted PSTATE.UAO has no impact on instruction execution until returning from the Illegal Execution state exception handler.

#### Workaround

No workaround is required for this erratum.

## 1256788

### Halting step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

#### Description

During Halting Step, execution of some load instructions in the Active-not-pending state might result in the execution of that instruction and the next instruction before returning control to the debugger by entering Debug state, instead of returning after a single instruction executed.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is in Halting Step mode.
2. The instruction being stepped is a load instruction that loads two or more destination registers.
3. Snoop invalidation of a cache line referenced by the load occurs during its execution, or an ECC error response occurs on the load.

#### Implications

If the above conditions are met, then two instructions can be stepped when a single step is expected, potentially resulting in unexpected DLR\_EL0 and DSPSR\_EL0 values upon entry to Debug state. However, the instructions still execute in the correct order and function correctly.

#### Workaround

There is no workaround for this erratum.

## 1264383

### Write-Back load after two Device-nG\* stores to the same physical address might get invalid data

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, and r3p0. Fixed in r3p1.

#### Description

In certain circumstances, a load to Write-Back memory might get a logical OR of two Device-nG\* stores to the same physical address. This does not happen with proper break-before-make page remapping, and only happens with two virtual addresses mapped to the same physical address and mismatched attributes. A data cache maintenance operation to this physical address between the stores and load to guarantee coherency also prevents this erratum. The load page translation needs to replace the store translation in the L1 data TLB, requiring accesses to 47 other pages in between.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Two stores to physical address A with Device-nG\* memory attribute occur.
2. Load/store accesses to 47 or more pages occur.
3. A load to physical address A with Write-Back memory attribute occurs.

#### Implications

If the above conditions are met, then under specific microarchitectural conditions, the load returns data that is a logical OR of the two or more stores.

#### Workaround

There is no workaround for this erratum.

## 1346756

### TLBI does not treat upper ASID bits as zero when TCR\_EL1.AS is 0

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, r3p1, and r4p0. Open.

#### Description

TLBI instructions are not treating ASID[15:8] as zero when TCR\_EL1.AS=0, as specified in the Arm Architecture Reference Manual. In this configuration, the bits are RES0, which should be written to zero by software, and ignored by hardware.

#### Configurations Affected

The erratum affects all configurations.

#### Conditions

1. TCR\_EL1.AS=0.
2. A TLBI is executed with ASID[15:8] not equal to zero.

#### Implications

The TLBI will execute locally and broadcast with an ASID that is out of range for this configuration.

#### Workaround

This erratum can be avoided if software is properly writing zero to RES0 bits.

## 1349291

### Uncontainable (UC) SError might be incorrectly logged as an Unrecoverable (UEU) SError

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

When an Uncontainable (UC) SError is reported or deferred by the core, it might be incorrectly logged as an Unrecoverable (UEU) SError. This is an inappropriate categorization downgrade which might allow for silent error propagation.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. An Uncontainable (UC) SError occurs in the system.
2. The Uncontainable (UC) SError is reported or deferred.

#### Implications

If the above conditions are met, then the ESR\_ELx.AET or DISR\_EL1.AET field might log the Uncontainable (UC) SError as an Unrecoverable (UEU) SError.

#### Workaround

This erratum can be mitigated by treating all SErrors reported with type Unrecoverable (UEU) as type Uncontainable (UC).

## 1356341

### L1D\_CACHE access related PMU events and L1D\_TLB access related PMU events increment on instructions/ micro-operations excluded from these events

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

The L1D\_CACHE access related PMU events 0x4, 0x40, and 0x41 and the L1D\_TLB access related PMU events 0x25, 0x4E, and 0x4F are incorrectly counting non-memory read/write operations that must be excluded. Software prefetch instructions are counted as read accesses and all other instructions are counted as write accesses.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

A software prefetch (PRFM) instruction or one of the following non-memory write operations is issued to the Load/Store Unit:

- A barrier (DMB, DSB, ESB, or PSB).
- A TLB Maintenance Operation (TMO).
- A Cache Maintenance Operation (CMO).
- An Address Translation operation (AT).
- A debug RAM read operation.

#### Implications

If any of the non-memory read/write operations listed above are issued to the Load/Store Unit, then the PMU counts for events L1D\_CACHE (0x4), L1D\_CACHE\_RD (0x40), L1D\_CACHE\_WR (0x41) or L1D\_TLB (0x25), L1D\_TLB\_RD (0x4E), and L1D\_TLB\_WR (0x4F) are incremented incorrectly.

#### Workaround

There is no workaround for this erratum.

## 1395332

### Read from PMCCNTR in AArch32 might return corrupted data

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

When PMCCNTR is configured to count core clock cycles, the result of a read from the PMCCNTR system register in AArch32 state might be corrupted. This corruption is predictable and occurs when the clock cycle count rolls over into the upper 32 bits of the register. For example, if PMCCNTR=0xFFFF\_FFFF and a read is executed around the time the clock cycle count is incremented, then the value returned might be 0x1\_FFFF\_FFFF rather than 0x1\_0000\_0000.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. PMCCNTR is configured to count core clock cycles.
2. The lower 32 bits of PMCCNTR contains a value close to 0xFFFF\_FFFF.
3. A read from PMCCNTR is performed in AArch32.

#### Implications

If the above conditions are met, then the read from the PMCCNTR register might return corrupted data.

#### Workaround

This erratum is not expected to require a workaround.

## 1406411

### MSR DSPSR\_EL0 while in debug state might not correctly update PSTATE.{N,C,Z,V,GE} on debug exit

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

An MSR DSPSR\_EL0 instruction that is executed in debug state and alters the Debug Saved Program Status Register, might fail to update PSTATE.{N,Z,C,V,GE} values on exit from debug state. This erratum applies to both AArch32 (MCR DSPSR) and AArch64 (MSR DSPSR\_EL0) operation.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core is in debug state.
2. The core executes an MSR instruction to alter the Debug Saved Program Status Register.
3. The core exits debug state.
4. The core might expose the incorrect PSTATE through execution of a conditional instruction or a read of PSTATE.{N,Z,C,V,GE} state.

#### Implications

If the above conditions are met, then this erratum might result in data corruption, incorrect program flow, or produce other undesirable effects. However, this erratum will not result in violation of access controls, for example, this erratum will not result in the core making accesses to Secure memory from Non-secure mode.

#### Workaround

The erratum can be avoided by setting CPUACTLR\_EL1[45] to 1 prior to exiting from debug state. Power consumption in the core will be higher when CPUACTLR\_EL1[45] is 1, as this prevents dynamic clock gating within sections of the core.

## 1408724

### Portions of the branch target address recorded in ETM trace information might be incorrect for some branches immediately preceding an indirect branch with a malformed branch target address

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

The errant behavior described in this erratum pertains solely to ETM reporting information, and strictly in close vicinity of ETM reporting of an indirect branch with a malformed branch target address (a programming error).

Information recorded in the ETM trace buffer for branch instructions includes the Virtual Address (VA) of the branch target. An indirect branch has a malformed branch target address when either the lowermost bits of the target address stipulate a misaligned instruction address, or the uppermost bits are non-canonical. Execution of an indirect branch with a malformed target address results in an Instruction Abort. ETM trace information correctly reports the malformed target address for the branch execution, and also correctly reports exception information for the Instruction Abort.

However, under rare circumstances, a few branches immediately preceding the indirect branch with malformed target address can incorrectly include the upper and lower portions of the malformed target address in the ETM trace information for the target of these earlier branches. Only the upper and lower portions of the branch target VAs are potentially mis-reported in the ETM trace information.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions:

1. ETM is enabled.
2. An indirect branch with a malformed branch target address is executed and traced.
3. Branch instructions immediately preceding the indirect branch with malformed target address are executed and traced.

#### Implications

If the above conditions are met, then within a tightly constrained window the branches immediately preceding the indirect branch with malformed target address might record partially corrupted target addresses in the ETM trace buffer.

#### Workaround

No workaround is required. The programming error should be evident to users from the ETM trace information pertaining to the indirect branch with a malformed branch target address and trace information from its resultant Instruction Abort.

## 1415323

### Ordering violation might occur when a load encounters an L1 tag RAM single bit ECC error when a snoop request targets the same line

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

If a core detects a false miss due to a single bit L1 tag RAM ECC error when executing a younger load instruction that bypassed another load or barrier and completed by forwarding data from a prior store instruction, then an ordering violation might occur in the presence of a snoop request.

#### Configurations Affected

The erratum affects all multi-core configurations with `CORE_CACHE_PROTECTION= 1`.

#### Conditions

1. Core A has a cache line X resident in the L1 data cache with write permissions, and has one or more stores in flight.
2. Core A performs a load (LD1) out-of-order for line X, bypassing another load or a barrier. The load encounters a tag single-bit ECC error, which makes the line appear as a miss, it allocates a miss request buffer requesting the line from L2.
3. LD1 is able to complete by forwarding data from an older store.
4. The older store drains and updates the L1 data cache.
5. Core B sends a snoop for line X and the snoop is ordered ahead of the miss request from LD1.
6. Core B performs a store to the line X.
7. Core A then receives the line X on behalf of its read request from LD1 and allocates the line.
8. Core A does not detect an ordering violation for the following:
  - An older load LD2 now observes this newer store, or
  - LD1 bypassed a load with acquire or barrier and is now required to observe the newer store.

#### Implications

If the above conditions are met, then under specific microarchitectural timing conditions, there might be an ordering violation, such as a read after read violation.

This has been graded as Programmer Category C because Arm expects this erratum to have a negligible impact over the undetected ECC failure rate in real systems. The reason for this categorization is that the issue only occurs during a very short window in time when using a highly implausible code sequence involving racing writes by multiple different cores.

#### Workaround

There is no workaround for this erratum.

## 1430754

### Write to External Debug Registers might cause a deadlock with certain AArch32 T32 code sequences

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, and r3p1. Fixed in r4p0.

#### Description

If a write to the External Debug Registers occurs such that it activates an address breakpoint, then under certain conditions the core might stop executing a few instructions before the breakpoint exception should occur.

#### Configurations Affected

This erratum affects all configurations with `CORE_CACHE_PROTECTION` set to `TRUE`.

#### Conditions

1. The core is executing in AArch32 T32 instruction state.
2. The breakpoint is set on a cacheable line.
3. The breakpoint is set on a cache line that starts with the final 16 bits of a 32-bit instruction.
4. There is a stuck-at-fault in the L1 instruction data array near the breakpoint location.
5. The breakpoint is activated using the External Debug Registers while the core is fetching.

#### Implications

If the above conditions are met, then the core might deadlock.

#### Workaround

Any interrupt will break the core out of the deadlock state.

## 1487185

### Waypoints from previous session might cause single-shot comparator match when trace enabled

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, r3p1, and r4p0. Open.

#### Description

On the first waypoint after the core ETM is enabled, it is possible for a single-shot comparator to have a spurious match based on the address from the last waypoint in the previous trace session.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

- The core ETM has been enabled, disabled, and re-enabled since the last reset.
- Single-shot address comparators are enabled.
- The last waypoint address before the core ETM was disabled either matches a single-shot comparator or causes a match in the range between waypoints depending on the single-shot control setup.

#### Implications

There might be a spurious single-shot comparator match, which might be used by the trace analyzer to activate other trace events.

#### Workaround

Between tracing sessions, set the core ETM to enter a prohibited region either instead of or in addition to disabling the ETM.

## 1490853

### TRCIDR3.CCITMIN value is incorrect

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r2p0, r3p0, r3p1, and r4p0. Open.

#### Description

Software reads of the TRCIDR3.CCITMIN field, corresponding to the instruction trace counting minimum threshold, observe the value 0x100 or a minimum cycle count threshold of 256. The correct value should be 0x4 for a minimum cycle count threshold of 4.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

- Software reads the TRCIDR3 ID register.
- Software uses the value of the CCITMIN field to determine minimum instruction trace cycle counting threshold to program the ETM.

#### Implications

If software uses the value returned by the TRCIDR3.CCITMIN field, then it will limit the range which could be used for programming the ETM. In reality, the ETM could be programmed with a much smaller value than what is indicated by the TRCIDR3.CCITMIN field and function correctly.

#### Workaround

The value for the TRCIDR3.CCITMIN field should be treated as 0x4.