



## Arm Cortex-A77 MP074

### Software Developer Errata Notice

Date of issue: March 10, 2025

Non-Confidential

Document version: 21.0

Copyright © 2025 Arm® Limited (or its affiliates). All rights reserved.

Document ID: SDEN-1152370

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



This document is Non-Confidential.

Copyright © 2025 Arm® Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.

Arm only permits use of this document if you have reviewed and accepted Arm's Proprietary notice found at the end of this document.

This document (SDEN\_1152370\_21.0\_en) was issued on March 10, 2025.

There might be a later issue at <http://developer.arm.com/documentation/SDEN-1152370>

## Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

If you find offensive language in this document, please email [terms@arm.com](mailto:terms@arm.com).

## Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm Cortex-A77 MP074, create a ticket on <https://support.developer.arm.com>.

To provide feedback on the document, fill the following survey:  
<https://developer.arm.com/documentation-feedback-survey>.

# Contents

<b>r0p0 implementation fixes</b>	8
<b>r1p0 implementation fixes</b>	8
<b>r1p1 implementation fixes</b>	8
<b>Introduction</b>	9
Scope	9
Categorization of errata	9
<b>Change Control</b>	10
<b>Errata summary table</b>	18
<b>Errata descriptions</b>	25
Category A	25
Category A (rare)	26
1316063 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	26
Category B	28
1160841 Continuous failing STREX because of another core snooping from speculatively executed atomic behind constantly mispredicted branch might cause livelock	28
1177367 Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation	29
1191167 MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result	30
1204882 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	31
1220737 Streaming store under specific conditions might cause deadlock or data corruption	33
1253791 Multiple floating-point divides/square roots concurrently completing back-to-back and flushing back-to-back might cause data corruption	34
1262841 Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation	35
1273521 A T32 instruction inside an IT block followed by a mispredicted speculative instruction stream might cause a deadlock	37
1450698 Software Step might prevent interrupt recognition	38
1467687 Branch prediction for an ERET cached in the instruction cache might cause a deadlock	40
1508412	41
1515815 The core might execute multiple instructions before taking a software step exception or halt step exception when the executing instruction resides in the L0 Macro-op cache	44

1791578	Atomic Store instructions to shareable write-back memory might cause memory consistency failures	45
Description		46
1800714	A transient single-bit ECC error in the MMU TC RAM might lead to stale translation in the L2 TLB	47
1852353	Watchpoint exception on Ld/St does not report correct address in FAR or EDWAR	48
1923201	External debugger access to Debug registers might not work during Warm reset	50
1925769	Store operation that encounters multiple hits in the TLB can access regions of memory with attributes that could not be accessed at that Exception level or Security state	51
1946167	Atomic instructions with acquire semantics might not be ordered with respect to older stores with release semantics	52
2356587	Continuous failing STREX because of another PE executing prefetch for store behind consistently mispredicted branch	54
2743100	The core might deadlock during powerdown sequence	55
3324348	MSR PSTATE.SSBS to 0 is not fully self-synchronizing	56
3439000	When Hardware Page Aggregation (HPA) is enabled memory accesses may be translated incorrectly	57
3696294	Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock	59
Category B (rare)		61
1286809	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	61
1418842	MRRC reads of some Generic Timer system registers in AArch32 mode might return corrupt data	63
1542418	The core might fetch a stale instruction from the L0 Macro-op cache which violates the ordering of instruction fetches	65
Category C		66
1148171	ERR0MISC0 might report incorrect BANK and SUBBANK values for transient parity errors in L1 instruction cache data array	66
1151664	Direct access to internal memory for L2 TLB might not update IDATAN_EL3 registers	67
1162083	16-bit T32 instruction close to breakpoint location may cause early breakpoint exception	68
1185469	Exception packet for return stack match might return incorrect [E1:E0] field	69
1192280	IMPLEMENTATION DEFINED fault for unsupported atomic operations is not routed to proper Exception level	70
1207839	Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error	71

1220404	Direct access to L1 data TLB might report incorrect value of valid bit of the corresponding TLB entry	72
1220843	ERROSTATUS.SERR encoding is incorrect for error responses from slave and deferred data errors from slave which are not supported	73
1244986	Illegal return event might corrupt PSTATE.UAO	74
1256789	Halting step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error	75
1262908	Write-Back load after two Device-nG* stores to the same physical address might get invalid data	76
1328683	Uncontainable (UC) SError might be incorrectly logged as an Unrecoverable (UEU) SError	77
1346768	TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0	78
1355135	L1D_CACHE access related PMU events and L1D_TLB access related PMU events increment on instructions/micro-operations excluded from these events	79
1395535	Read from PMCCNTR in AArch32 might return corrupted data	80
1405548	MSR DSPSR_ELO while in debug state might not correctly update PSTATE. {N,C,Z,V,GE} on debug exit	81
1415321	LDREX-STREX might succeed incorrectly when an intervening store occurs and LDREX detects a single-bit ECC error on the cache line in the L1 data cache tag RAM	82
1421023	Portions of the branch target address recorded in ETM trace information are incorrect for an indirect branch with a malformed branch target address	84
1487187	Waypoints from previous session might cause single-shot comparator match when trace enabled	85
1488613	An unaligned load might initiate a prefetch request which crosses a page boundary	86
1491015	TRCIDR3.CCITMIN value is incorrect	87
1514033	Error Synchronization Barrier (ESB) instruction execution with a pending masked Virtual SError might not clear HCR_EL2.VSE	88
1519163	AMU Counter INST_RETIRED does not increment correctly when 16 instructions retire in same cycle	89
1522097	The core might detect a breakpoint exception one instruction earlier than the programmed location when the L0 Macro-op cache contains an instruction that is affected by a parity error	90
1523503	CPUECTLR_EL1 controls for the MMU have no affect	91
1610369	ERR0MISC0_EL1.SUBARRAY value for ECC errors in the L1 data cache might be incorrect	92
1624431	CPUAMEVTYPER4_ELO register cannot be written	93
1655748	MRC read of DBGDSCRint into APSR_nzcv might produce wrong results and lead to corruption	94
1662411	Executing a cache maintenance by set/way instruction targeting the L1 data cache in the presence of snoops might result in a deadlock	95

1702492	The core might not update IDATA*_EL3 correctly by a direct memory access to L1 Instruction Cache Tag or L1 Instruction TLB	97
1788065	Possible loss of CTI event	98
1788067	Loss of CTI events during warm reset	99
1791402	The core might deadlock when an external debugger injects instructions using ITR register	100
1827134	External debug accesses in memory access mode with SCTLRL_ELx.IESB set might result in unpredictable behavior	101
1830646	Watchpoint Exception on DC ZVA does not report correct address in FAR or EDWAR	102
1857204	A memory mapped write to PMSSRR might falsely cause some PMU counters and counter overflow status to be reset after snapshot capture and read might return unknown/written data	103
1857621	Uncorrectable tag errors in L2 cache might cause deadlock	104
1869877	ERR0MISC0_EL1.SUBARRAY, ERR0STATUS.CE and ERR0STATUS.DE values for ECC errors in the L1 data cache might be incorrect	105
1880115	Noncompliance with prioritization of Exception Catch debug events	106
1884878	The core might report incorrect fetch address to FAR_ELx when the core is fetching an instruction from a virtual address associated with a page table entry which has been modified	108
1885949	L2 data RAM may fail to report corrected ECC errors	109
1899210	Some corrected errors might incorrectly increment ERR0MISC0.CECR or ERR0MISC0.CECO	110
1899434	PFG duplicate reported faults through a Warm reset	111
1923198	IDATAN_EL3 might represent incorrect value after direct memory access to internal memory for Instruction TLB	112
1950447	The PE might deadlock if Pseudofault Injection is enabled in Debug State	113
2001419	DRPS might not execute correctly in Debug state with SCTLRL_ELx.IESB set in the current EL	114
2033523	ETM trace information records a branch to the next instruction as an N atom	115
2052427	An execution of MSR instruction might not update the destination register correctly when an external debugger initiates an APB write operation to update debug registers	116
2110734	External APB write to a register located at offset 0x084 might incorrectly issue a write to External Debug Instruction Transfer Register	118
2141649	A64 WFI or A64 WFE executed in Debug state suspends execution indefinitely	120
2238113	Reads of DISR_EL1 incorrectly return 0s while in Debug State	122
2239142	DRPS instruction is not treated as UNDEFINED at EL0 in Debug state	123
2263696	L1 Data poison is not cleared by a store	124
2278130	PMU L1D_CACHE_REFILL_OUTER is inaccurate	125

2307836	ESR_ELx.ISV can be set incorrectly for an external abort on translation table walk	126
2391682	Software-step not done after exit from Debug state with an illegal value in DSPSR	127
2816903	PE might fail to detect multiple uncorrectable ECC errors in the L1 data cache tag RAM	128
3049877	L1D_TLB access related PMU event increments more than once per memory access	129
3605050	Incorrect count for PMU event 0x004C (L1D_TLB_REFILL_RD) might be observed	130
3607348	PSTATE.{PAN,UAO} synchronization might not be honored while MSR PSTATE is speculative	132
3633467	EDSCR.STATUS not updated on Halting Step when a Load-Exclusive instruction generates a synchronous exception	133
3700181	PE might fail to log a RAS error for L2 data RAM ECC errors	134
3705917	PMU events are mis-categorized by not considering the effect of "Taken locally"	135
<b>Proprietary notice</b>		136
<b>Product and document information</b>		138
Product status		138
Product completeness status		138
Product revision status		138

## r0p0 implementation fixes

Note the following errata might be fixed in some implementations of r0p0. 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]	1220737 Streaming store under specific conditions might cause deadlock or data corruption
---------------	---

Note that there is no change to the MIDR\_EL1 which remains at r0p0 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.

## 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]	1316063 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.
---------------	--

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.

## r1p1 implementation fixes

Note the following errata might be fixed in some implementations of r1p1. 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[3]	1450698 Software Step might prevent interrupt recognition
---------------	---

Note that there is no change to the MIDR\_EL1 which remains at r1p1 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:

<b>Category A</b>	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
<b>Category A (Rare)</b>	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.
<b>Category B</b>	A significant error or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
<b>Category B (Rare)</b>	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.
<b>Category C</b>	A minor error.

# Change Control

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

## March 10, 2025: Changes in document version v21.0

ID	Status	Area	Category	Summary
<a href="#">3439000</a>	New	Programmer	Category B	When Hardware Page Aggregation (HPA) is enabled memory accesses may be translated incorrectly

## October 01, 2024: Changes in document version v20.0

ID	Status	Area	Category	Summary
<a href="#">3696294</a>	New	Programmer	Category B	Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock
<a href="#">3605050</a>	New	Programmer	Category C	Incorrect count for PMU event 0x004C (L1D_TLB_REFILL_RD) might be observed
<a href="#">3607348</a>	New	Programmer	Category C	PSTATE.{PAN,UAO} synchronization might not be honored while MSR PSTATE is speculative
<a href="#">3633467</a>	New	Programmer	Category C	EDSCR.STATUS not updated on Halting Step when a Load-Exclusive instruction generates a synchronous exception
<a href="#">3700181</a>	New	Programmer	Category C	PE might fail to log a RAS error for L2 data RAM ECC errors
<a href="#">3705917</a>	New	Programmer	Category C	PMU events are mis-categorized by not considering the effect of "Taken locally"

## May 09, 2024: Changes in document version v19.0

ID	Status	Area	Category	Summary
<a href="#">3324348</a>	New	Programmer	Category B	MSR PSTATE.SBS to 0 is not fully self-synchronizing

## September 28, 2023: Changes in document version v18.0

ID	Status	Area	Category	Summary
<a href="#">1286809</a>	Updated	Programmer	Category B (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
<a href="#">3049877</a>	New	Programmer	Category C	L1D_TLB PMU event increments more than once per memory access

## January 31, 2023: Changes in document version v17.0

ID	Status	Area	Category	Summary
<a href="#">1262841</a>	Updated	Programmer	Category B	Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation
<a href="#">2816903</a>	New	Programmer	Category C	PE might fail to detect multiple uncorrectable ECC errors in the L1 data cache tag RAM

**October 20, 2022: Changes in document version v16.0**

ID	Status	Area	Category	Summary
<a href="#">1262841</a>	Updated	Programmer	Category B	Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation
<a href="#">2743100</a>	New	Programmer	Category B	The core might deadlock during powerdown sequence

**March 04, 2022: Changes in document version v15.0**

ID	Status	Area	Category	Summary
<a href="#">2356587</a>	New	Programmer	Category B	Continuous failing STREX because of another PE executing prefetch for store behind consistently mispredicted branch
<a href="#">2307836</a>	New	Programmer	Category C	ESR_ELx.ISV can be set incorrectly for an external abort on translation table walk
<a href="#">2391682</a>	New	Programmer	Category C	Software-step not done after exit from Debug state with an illegal value in DSPSR

**August 26, 2021: Changes in document version v14.0**

ID	Status	Area	Category	Summary
<a href="#">2141649</a>	New	Programmer	Category C	A64 WFI or A64 WFE executed in Debug state suspends execution indefinitely
<a href="#">2238113</a>	New	Programmer	Category C	Reads of DISR_EL1 incorrectly return 0s while in Debug State
<a href="#">2239142</a>	New	Programmer	Category C	DRPS instruction is not treated as UNDEFINED at EL0 in Debug state
<a href="#">2263696</a>	New	Programmer	Category C	L1 Data poison is not cleared by a store
<a href="#">2278130</a>	New	Programmer	Category C	PMU L1D_CACHE_REFILL_OUTER is inaccurate

**April 06, 2021: Changes in document version v13.0**

ID	Status	Area	Category	Summary
<a href="#">1791578</a>	New	Programmer	Category B	Atomic Store instructions to shareable write-back memory might cause memory consistency failures
<a href="#">1852353</a>	New	Programmer	Category B	Watchpoint exception on Ld/St does not report correct address in FAR or EDWAR
<a href="#">1655748</a>	New	Programmer	Category C	MRC read of DBGDSCRint into APSR_nzcv might produce wrong results and lead to corruption
<a href="#">1791402</a>	New	Programmer	Category C	The core might deadlock when an external debugger injects instructions using ITR register
<a href="#">1857621</a>	New	Programmer	Category C	Uncorrectable tag errors in L2 cache might cause deadlock
<a href="#">1885949</a>	New	Programmer	Category C	L2 data RAM may fail to report corrected ECC errors
<a href="#">2033523</a>	New	Programmer	Category C	ETM trace information records a branch to the next instruction as an N atom
<a href="#">2052427</a>	New	Programmer	Category C	An execution of MSR instruction might not update the destination register correctly when an external debugger initiates an APB write operation to update debug registers
<a href="#">2110734</a>	New	Programmer	Category C	External APB write to a register located at offset 0x084 might incorrectly issue a write to External Debug Instruction Transfer Register

## November 06, 2020: Changes in document version v12.0

ID	Status	Area	Category	Summary
<a href="#">1946167</a>	New	Programmer	Category B	Atomic instructions with acquire semantics might not be ordered with respect to older stores with release semantics
<a href="#">1950447</a>	New	Programmer	Category C	The PE might deadlock if Pseudofault Injection is enabled in Debug State
<a href="#">2001419</a>	New	Programmer	Category C	DRPS might not execute correctly in Debug state with SCTLR_ElX.IESB set in the current EL

## September 01, 2020: Changes in document version v11.0

ID	Status	Area	Category	Summary
<a href="#">1508412</a>	Updated	Programmer	Category B	NC/Device Load and Store Exclusive or PAR-Read collision can cause deadlock
<a href="#">1923201</a>	New	Programmer	Category B	External debugger access to Debug registers might not work during Warm reset
<a href="#">1925769</a>	New	Programmer	Category B	Store operation that encounters multiple hits in the TLB might access regions of memory with attributes that could not be accessed at that exception level or security state
<a href="#">1857204</a>	New	Programmer	Category C	A memory mapped write to PMSSRR might falsely cause some PMU counters and counter overflow status to be reset after snapshot capture and read might return unknown/written data
<a href="#">1869877</a>	New	Programmer	Category C	ERRORMISCO_EL1.SUBARRAY, ERROSTATUS.CE and ERROSTATUS.DE values for ECC errors in the L1 data cache might be incorrect
<a href="#">1880115</a>	New	Programmer	Category C	Noncompliance with prioritization of Exception Catch debug events
<a href="#">1884878</a>	New	Programmer	Category C	The core might report incorrect fetch address to FAR_ELn when the core is fetching an instruction from a virtual address associated with a page table entry which has been modified
<a href="#">1899210</a>	New	Programmer	Category C	Some corrected errors might incorrectly increment ERRORMISCO.CECR or ERRORMISCO.CECO
<a href="#">1899434</a>	New	Programmer	Category C	PFG duplicate reported faults through a warm reset
<a href="#">1923198</a>	New	Programmer	Category C	IDATAN_EL3 might represent incorrect value after direct memory access to internal memory for Instruction TLB

## May 19, 2020: Changes in document version v10.0

ID	Status	Area	Category	Summary
<a href="#">1508412</a>	Updated	Programmer	Category B	NC/Device Load and Store Exclusive or PAR-Read collision can cause deadlock
<a href="#">1800714</a>	New	Programmer	Category B	A transient single-bit ECC error in the MMU TC RAM might lead to stale translation in the L2 TLB
<a href="#">1662411</a>	New	Programmer	Category C	Executing a cache maintenance by set/way instruction targeting the L1 data cache in the presence of snoops might result in a deadlock
<a href="#">1702492</a>	New	Programmer	Category C	The core might not update IDATA*_EL3 correctly by a direct memory access to L1 Instruction Cache Tag or L1 Instruction TLB
<a href="#">1788065</a>	New	Programmer	Category C	Possible loss of CTI event
<a href="#">1788067</a>	New	Programmer	Category C	Loss of CTI events during warm reset
<a href="#">1827134</a>	New	Programmer	Category C	External debug accesses in memory access mode with SCTLR_ELx.IESB set might result in unpredictable behavior
<a href="#">1830646</a>	New	Programmer	Category C	Watchpoint Exception on DC ZVA does not report correct address in FAR or EDWAR

## November 08, 2019: Changes in document version v9.0

ID	Status	Area	Category	Summary
<a href="#">1467687</a>	Updated	Programmer	Category B	Branch prediction for an ERET cached in the instruction cache might cause a deadlock
<a href="#">1508412</a>	Updated	Programmer	Category B	NC/Device Load and Store Exclusive or PAR-Read collision can cause deadlock
<a href="#">1542418</a>	New	Programmer	Category B (rare)	The core might fetch a stale instruction from the L0 Macro-op cache which violates the ordering of instruction fetches
<a href="#">1514033</a>	New	Programmer	Category C	Error Synchronization Barrier (ESB) instruction execution with a pending masked Virtual SError might not clear HCR_EL2.VSE
<a href="#">1519163</a>	New	Programmer	Category C	AMU Counter INST_RETIRED does not increment correctly when 16 instructions retire in same cycle
<a href="#">1522097</a>	New	Programmer	Category C	The core might detect a breakpoint exception one instruction earlier than the programmed location when the L0 Macro-op cache contains an instruction that is affected by a parity error
<a href="#">1523503</a>	New	Programmer	Category C	CPUECTLR_EL1 controls for the MMU have no affect
<a href="#">1610369</a>	New	Programmer	Category C	ERRORMISC0_EL1.SUBARRAY value for ECC errors in the L1 data cache might be incorrect
<a href="#">1624431</a>	New	Programmer	Category C	CPUAMEVTYPEPER4_ELO register cannot be written

**July 15, 2019: Changes in document version v8.0**

ID	Status	Area	Category	Summary
<a href="#">1508412</a>	New	Programmer	Category B	NC/Device Load and Store Exclusive or PAR-Read collision can cause deadlock
<a href="#">1515815</a>	New	Programmer	Category B	The core might execute multiple instructions before taking a software step exception or halt step exception when the executing instruction resides in the L0 Macro-op cache
<a href="#">1487187</a>	New	Programmer	Category C	Waypoints from previous session might cause single-shot comparator match when trace enabled
<a href="#">1488613</a>	New	Programmer	Category C	An unaligned load might initiate a prefetch request which crosses a page boundary
<a href="#">1491015</a>	New	Programmer	Category C	TRCIDR3.CCITMIN value is incorrect

**May 24, 2019: Changes in document version v7.0**

ID	Status	Area	Category	Summary
<a href="#">1450698</a>	New	Programmer	Category B	Software Step might prevent interrupt recognition
<a href="#">1467687</a>	New	Programmer	Category B	Branch prediction for an ERET cached in the instruction cache might cause a deadlock

**March 29, 2019: Changes in document version v6.0**

ID	Status	Area	Category	Summary
<a href="#">1418842</a>	New	Programmer	Category B (rare)	MRRC reads of some Generic Timer system registers in AArch32 mode might return corrupt data
<a href="#">1328683</a>	New	Programmer	Category C	Uncontainable (UC) SError might be incorrectly logged as an Unrecoverable (UEU) SError
<a href="#">1346768</a>	New	Programmer	Category C	TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0
<a href="#">1355135</a>	New	Programmer	Category C	L1D_CACHE access related PMU events and L1D_TLB access related PMU events increment on instructions/micro-operations excluded from these events
<a href="#">1395535</a>	New	Programmer	Category C	Read from PMCCNTR in AArch32 might return corrupted data
<a href="#">1405548</a>	New	Programmer	Category C	MSR DSPSR_ELO while in debug state might not correctly update PSTATE.{N,C,Z,V,GE} on debug exit
<a href="#">1415321</a>	New	Programmer	Category C	LDREX-STREX might succeed incorrectly when an intervening store occurs and LDREX detects a single-bit ECC error on the cache line in the L1 data cache tag RAM
<a href="#">1421023</a>	New	Programmer	Category C	Portions of the branch target address recorded in ETM trace information are incorrect for an indirect branch with a malformed branch target address

**March 15, 2019: Changes in document version v5.0**

No new or updated errata in this document version.

## December 03, 2018: Changes in document version v4.0

ID	Status	Area	Category	Summary
<a href="#">1316063</a>	New	Programmer	Category A (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
<a href="#">1286809</a>	New	Programmer	Category B (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

## October 05, 2018: Changes in document version v3.0

ID	Status	Area	Category	Summary
<a href="#">1160841</a>	Updated	Programmer	Category B	Continuous failing STREX because of another core snooping from speculatively executed atomic behind constantly mispredicted branch might cause livelock
<a href="#">1177367</a>	Updated	Programmer	Category B	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation Status
<a href="#">1191167</a>	Updated	Programmer	Category B	MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result
<a href="#">1204882</a>	Updated	Programmer	Category B	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
<a href="#">1220737</a>	New	Programmer	Category B	Streaming store under specific conditions might cause deadlock or data corruption
<a href="#">1253791</a>	New	Programmer	Category B	Multiple floating-point divides/square roots concurrently completing back-to-back and flushing back-to-back might cause data corruption
<a href="#">1262841</a>	New	Programmer	Category B	Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation
<a href="#">1273521</a>	New	Programmer	Category B	A T32 instruction inside an IT block followed by a mispredicted speculative instruction stream might cause a deadlock
<a href="#">1148171</a>	Updated	Programmer	Category C	ERR0MISC0 might report incorrect BANK and SUBBANK values for transient parity errors in L1 instruction cache data array
<a href="#">1151664</a>	Updated	Programmer	Category C	Direct access to internal memory for L2 TLB might not update IDATAn_EL3 registers
<a href="#">1162083</a>	Updated	Programmer	Category C	16-bit T32 instruction close to breakpoint location may cause early breakpoint exception
<a href="#">1185469</a>	Updated	Programmer	Category C	Exception packet for return stack match might return incorrect [E1:E0] field
<a href="#">1192280</a>	Updated	Programmer	Category C	IMPLEMENTATION DEFINED fault for unsupported atomic operations is not routed to proper Exception level
<a href="#">1207839</a>	Updated	Programmer	Category C	Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error
<a href="#">1220404</a>	New	Programmer	Category C	Direct access to L1 data TLB might report incorrect value of valid bit of the corresponding TLB entry
<a href="#">1220843</a>	New	Programmer	Category C	ERR0STATUS.SERR encoding is incorrect for error responses from slave and deferred data errors from slave which are not supported
<a href="#">1244986</a>	New	Programmer	Category C	Illegal return event might corrupt PSTATE.UAO
<a href="#">1256789</a>	New	Programmer	Category C	Halting step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error
<a href="#">1262908</a>	New	Programmer	Category C	Write-Back load after two Device-nG* stores to the same physical address might get invalid data



## July 27, 2018: Changes in document version v2.0

ID	Status	Area	Category	Summary
<a href="#">1160841</a>	New	Programmer	Category B	Continuous failing STREX because of another core snooping from speculatively executed atomic behind constantly mispredicted branch might cause livelock
<a href="#">1177367</a>	New	Programmer	Category B	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation Status
<a href="#">1191167</a>	New	Programmer	Category B	MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result
<a href="#">1204882</a>	New	Programmer	Category B	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
<a href="#">1162083</a>	New	Programmer	Category C	16-bit T32 instruction close to breakpoint location may cause early breakpoint exception
<a href="#">1185469</a>	New	Programmer	Category C	Exception packet for return stack match might return incorrect [E1:E0] field
<a href="#">1192280</a>	New	Programmer	Category C	IMPLEMENTATION DEFINED fault for unsupported atomic operations is not routed to proper Exception level
<a href="#">1207839</a>	New	Programmer	Category C	Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error

## May 16, 2018: Changes in document version v1.0

ID	Status	Area	Category	Summary
<a href="#">1148171</a>	New	Programmer	Category C	ERR0MISC0 might report incorrect BANK and SUBBANK values for transient parity errors in L1 instruction cache data array
<a href="#">1151664</a>	New	Programmer	Category C	Direct access to internal memory for L2 TLB might not update IDATAN_EL3 registers

# Errata summary table

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

ID	Area	Category	Summary	Found in versions	Fixed in version
<a href="#">1316063</a>	Programmer	Category A (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	r1p1
<a href="#">1160841</a>	Programmer	Category B	Continuous failing STREX because of another core snooping from speculatively executed atomic behind constantly mispredicted branch might cause livelock	r0p0	r1p0
<a href="#">1177367</a>	Programmer	Category B	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation Status	r0p0	r1p0
<a href="#">1191167</a>	Programmer	Category B	MRC read following MRRC read of specific Generic Timer in AArch32 might give incorrect result	r0p0	r1p0
<a href="#">1204882</a>	Programmer	Category B	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
<a href="#">1220737</a>	Programmer	Category B	Streaming store under specific conditions might cause deadlock or data corruption	r0p0	r1p0
<a href="#">1253791</a>	Programmer	Category B	Multiple floating-point divides/square roots concurrently completing back-to-back and flushing back-to-back might cause data corruption	r0p0	r1p0
<a href="#">1262841</a>	Programmer	Category B	Translation access hitting a prefetched L2 TLB entry under specific conditions might corrupt the L2 TLB leading to an incorrect translation	r0p0	r1p0
<a href="#">1273521</a>	Programmer	Category B	A T32 instruction inside an IT block followed by a mispredicted speculative instruction stream might cause a deadlock	r0p0	r1p0
<a href="#">1450698</a>	Programmer	Category B	Software Step might prevent interrupt recognition	r0p0, r1p0	r1p1

ID	Area	Category	Summary	Found in versions	Fixed in version
<a href="#">1467687</a>	Programmer	Category B	Branch prediction for an ERET cached in the instruction cache might cause a deadlock	r0p0, r1p0	r1p1
<a href="#">1508412</a>	Programmer	Category B	NC/Device Load and Store Exclusive or PAR-Read collision can cause deadlock	r0p0, r1p0	r1p1
<a href="#">1515815</a>	Programmer	Category B	The core might execute multiple instructions before taking a software step exception or halt step exception when the executing instruction resides in the L0 Macro-op cache	r0p0, r1p0	r1p1
<a href="#">1791578</a>	Programmer	Category B	Atomic Store instructions to shareable write-back memory might cause memory consistency failures	r0p0, r1p0, r1p1	Open
<a href="#">1800714</a>	Programmer	Category B	A transient single-bit ECC error in the MMU TC RAM might lead to stale translation in the L2 TLB	r0p0, r1p0, r1p1	Open
<a href="#">1852353</a>	Programmer	Category B	Watchpoint exception on Ld/St does not report correct address in FAR or EDWAR	r0p0, r1p0, r1p1	Open
<a href="#">1923201</a>	Programmer	Category B	External debugger access to Debug registers might not work during Warm reset	r0p0, r1p0, r1p1	Open
<a href="#">1925769</a>	Programmer	Category B	Store operation that encounters multiple hits in the TLB might access regions of memory with attributes that could not be accessed at that exception level or security state	r0p0, r1p0, r1p1	Open
<a href="#">1946167</a>	Programmer	Category B	Atomic instructions with acquire semantics might not be ordered with respect to older stores with release semantics	r0p0, r1p0, r1p1	Open
<a href="#">2356587</a>	Programmer	Category B	Continuous failing STREX because of another PE executing prefetch for store behind consistently mispredicted branch	r0p0, r1p0, r1p1	Open
<a href="#">2743100</a>	Programmer	Category B	The core might deadlock during powerdown sequence	r0p0, r1p0, r1p1	Open
<a href="#">3324348</a>	Programmer	Category B	MSR PSTATE.SSBS to 0 is not fully self-synchronizing	r0p0, r1p0, r1p1	Open
<a href="#">3439000</a>	Programmer	Category B	When Hardware Page Aggregation (HPA) is enabled memory accesses may be translated incorrectly	r0p0, r1p0, r1p1	Open

ID	Area	Category	Summary	Found in versions	Fixed in version
<a href="#">3696294</a>	Programmer	Category B	Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock	r0p0, r1p0, r1p1	Open
<a href="#">1286809</a>	Programmer	Category B (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	r1p1
<a href="#">1418842</a>	Programmer	Category B (rare)	MRRC reads of some Generic Timer system registers in AArch32 mode might return corrupt data	r0p0, r1p0	r1p1
<a href="#">1542418</a>	Programmer	Category B (rare)	The core might fetch a stale instruction from the L0 Macro-op cache which violates the ordering of instruction fetches	r0p0, r1p0	r1p1
<a href="#">1148171</a>	Programmer	Category C	ERR0MISC0 might report incorrect BANK and SUBBANK values for transient parity errors in L1 instruction cache data array	r0p0	r1p0
<a href="#">1151664</a>	Programmer	Category C	Direct access to internal memory for L2 TLB might not update IDATAn_EL3 registers	r0p0	r1p0
<a href="#">1162083</a>	Programmer	Category C	16-bit T32 instruction close to breakpoint location may cause early breakpoint exception	r0p0	r1p0
<a href="#">1185469</a>	Programmer	Category C	Exception packet for return stack match might return incorrect [E1:E0] field	r0p0	r1p0
<a href="#">1192280</a>	Programmer	Category C	IMPLEMENTATION DEFINED fault for unsupported atomic operations is not routed to proper Exception level	r0p0	r1p0
<a href="#">1207839</a>	Programmer	Category C	Software step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error	r0p0	r1p0
<a href="#">1220404</a>	Programmer	Category C	Direct access to L1 data TLB might report incorrect value of valid bit of the corresponding TLB entry	r0p0	r1p0
<a href="#">1220843</a>	Programmer	Category C	ERROSTATUS.SERR encoding is incorrect for error responses from slave and deferred data errors from slave which are not supported	r0p0	r1p0
<a href="#">1244986</a>	Programmer	Category C	Illegal return event might corrupt PSTATE.UAO	r0p0	r1p0

ID	Area	Category	Summary	Found in versions	Fixed in version
<a href="#">1256789</a>	Programmer	Category C	Halting step might see extra instruction executed for some loads when crossed with snoop invalidation or ECC error	r0p0	r1p0
<a href="#">1262908</a>	Programmer	Category C	Write-Back load after two Device-nG* stores to the same physical address might get invalid data	r0p0	r1p0
<a href="#">1328683</a>	Programmer	Category C	Uncontainable (UC) SError might be incorrectly logged as an Unrecoverable (UEU) SError	r0p0, r1p0	r1p1
<a href="#">1346768</a>	Programmer	Category C	TLBI does not treat upper ASID bits as zero when TCR_EL1.AS is 0	r0p0, r1p0, r1p1	Open
<a href="#">1355135</a>	Programmer	Category C	L1D_CACHE access related PMU events and L1D_TLB access related PMU events increment on instructions/micro-operations excluded from these events	r0p0, r1p0	r1p1
<a href="#">1395535</a>	Programmer	Category C	Read from PMCCNTR in AArch32 might return corrupted data	r0p0, r1p0	r1p1
<a href="#">1405548</a>	Programmer	Category C	MSR DSPSR_ELO while in debug state might not correctly update PSTATE.{N,C,Z,V,GE} on debug exit	r0p0, r1p0	r1p1
<a href="#">1415321</a>	Programmer	Category C	LDREX-STREX might succeed incorrectly when an intervening store occurs and LDREX detects a single-bit ECC error on the cache line in the L1 data cache tag RAM	r0p0, r1p0	r1p1
<a href="#">1421023</a>	Programmer	Category C	Portions of the branch target address recorded in ETM trace information are incorrect for an indirect branch with a malformed branch target address	r0p0, r1p0	r1p1
<a href="#">1487187</a>	Programmer	Category C	Waypoints from previous session might cause single-shot comparator match when trace enabled	r0p0, r1p0	r1p1
<a href="#">1488613</a>	Programmer	Category C	An unaligned load might initiate a prefetch request which crosses a page boundary	r0p0, r1p0	r1p1
<a href="#">1491015</a>	Programmer	Category C	TRCIDR3.CCITMIN value is incorrect	r0p0, r1p0	r1p1
<a href="#">1514033</a>	Programmer	Category C	Error Synchronization Barrier (ESB) instruction execution with a pending masked Virtual SError might not clear HCR_EL2.VSE	r0p0, r1p0	r1p1
<a href="#">1519163</a>	Programmer	Category C	AMU Counter INST_RETIRED does not increment correctly when 16 instructions retire in same cycle	r0p0, r1p0	r1p1

ID	Area	Category	Summary	Found in versions	Fixed in version
<a href="#">1522097</a>	Programmer	Category C	The core might detect a breakpoint exception one instruction earlier than the programmed location when the L0 Macro-op cache contains an instruction that is affected by a parity error	r0p0, r1p0	r1p1
<a href="#">1523503</a>	Programmer	Category C	CPUECTLR_EL1 controls for the MMU have no affect	r0p0, r1p0	r1p1
<a href="#">1610369</a>	Programmer	Category C	ERR0MISCO_EL1.SUBARRAY value for ECC errors in the L1 data cache might be incorrect	r0p0, r1p0, r1p1	Open
<a href="#">1624431</a>	Programmer	Category C	CPUAMEVTPYPER4_EL0 register cannot be written	r0p0, r1p0	r1p1
<a href="#">1655748</a>	Programmer	Category C	MRC read of DBGDSCRint into APSR_nzcv might produce wrong results and lead to corruption	r0p0, r1p0, r1p1	Open
<a href="#">1662411</a>	Programmer	Category C	Executing a cache maintenance by set/way instruction targeting the L1 data cache in the presence of snoops might result in a deadlock	r0p0, r1p0, r1p1	Open
<a href="#">1702492</a>	Programmer	Category C	The core might not update IDATA*_EL3 correctly by a direct memory access to L1 Instruction Cache Tag or L1 Instruction TLB	r0p0, r1p0, r1p1	Open
<a href="#">1788065</a>	Programmer	Category C	Possible loss of CTI event	r0p0, r1p0, r1p1	Open
<a href="#">1788067</a>	Programmer	Category C	Loss of CTI events during warm reset	r0p0, r1p0, r1p1	Open
<a href="#">1791402</a>	Programmer	Category C	The core might deadlock when an external debugger injects instructions using ITR register	r0p0, r1p0, r1p1	Open
<a href="#">1827134</a>	Programmer	Category C	External debug accesses in memory access mode with SCTLR_ELx.IESB set might result in unpredictable behavior	r0p0, r1p0, r1p1	Open
<a href="#">1830646</a>	Programmer	Category C	Watchpoint Exception on DC ZVA does not report correct address in FAR or EDWAR	r0p0, r1p0, r1p1	Open
<a href="#">1857204</a>	Programmer	Category C	A memory mapped write to PMSSRR might falsely cause some PMU counters and counter overflow status to be reset after snapshot capture and read might return unknown/written data	r0p0, r1p0, r1p1	Open
<a href="#">1857621</a>	Programmer	Category C	Uncorrectable tag errors in L2 cache might cause deadlock	r0p0, r1p0, r1p1	Open

ID	Area	Category	Summary	Found in versions	Fixed in version
<a href="#">1869877</a>	Programmer	Category C	ERRORMISCO_EL1.SUBARRAY, ERRORSTATUS.CE and ERRORSTATUS.DE values for ECC errors in the L1 data cache might be incorrect	r0p0, r1p0, r1p1	Open
<a href="#">1880115</a>	Programmer	Category C	Noncompliance with prioritization of Exception Catch debug events	r0p0, r1p0, r1p1	Open
<a href="#">1884878</a>	Programmer	Category C	The core might report incorrect fetch address to FAR_ELn when the core is fetching an instruction from a virtual address associated with a page table entry which has been modified	r0p0, r1p0, r1p1	Open
<a href="#">1885949</a>	Programmer	Category C	L2 data RAM may fail to report corrected ECC errors	r0p0, r1p0, r1p1	Open
<a href="#">1899210</a>	Programmer	Category C	Some corrected errors might incorrectly increment ERRORMISCO.CECR or ERRORMISCO.CECO	r0p0, r1p0, r1p1	Open
<a href="#">1899434</a>	Programmer	Category C	PFG duplicate reported faults through a warm reset	r0p0, r1p0, r1p1	Open
<a href="#">1923198</a>	Programmer	Category C	IDATAN_EL3 might represent incorrect value after direct memory access to internal memory for Instruction TLB	r0p0, r1p0, r1p1	Open
<a href="#">1950447</a>	Programmer	Category C	The PE might deadlock if Pseudofault Injection is enabled in Debug State	r0p0, r1p0, r1p1	Open
<a href="#">2001419</a>	Programmer	Category C	DRPS might not execute correctly in Debug state with SCTLR_ELx.IESB set in the current EL	r0p0, r1p0, r1p1	Open
<a href="#">2033523</a>	Programmer	Category C	ETM trace information records a branch to the next instruction as an N atom	r0p0, r1p0, r1p1	Open
<a href="#">2052427</a>	Programmer	Category C	An execution of MSR instruction might not update the destination register correctly when an external debugger initiates an APB write operation to update debug registers	r0p0, r1p0, r1p1	Open
<a href="#">2110734</a>	Programmer	Category C	External APB write to a register located at offset 0x084 might incorrectly issue a write to External Debug Instruction Transfer Register	r0p0, r1p0, r1p1	Open
<a href="#">2141649</a>	Programmer	Category C	A64 WFI or A64 WFE executed in Debug state suspends execution indefinitely	r0p0, r1p0, r1p1	Open

ID	Area	Category	Summary	Found in versions	Fixed in version
<a href="#">2238113</a>	Programmer	Category C	Reads of DISR_EL1 incorrectly return 0s while in Debug State	r0p0, r1p0, r1p1	Open
<a href="#">2239142</a>	Programmer	Category C	DRPS instruction is not treated as UNDEFINED at ELO in Debug state	r0p0, r1p0, r1p1	Open
<a href="#">2263696</a>	Programmer	Category C	L1 Data poison is not cleared by a store	r0p0, r1p0, r1p1	Open
<a href="#">2278130</a>	Programmer	Category C	PMU L1D_CACHE_REFILL_OUTER is inaccurate	r0p0, r1p0, r1p1	Open
<a href="#">2307836</a>	Programmer	Category C	ESR_ELx.ISV can be set incorrectly for an external abort on translation table walk	r0p0, r1p0, r1p1	Open
<a href="#">2391682</a>	Programmer	Category C	Software-step not done after exit from Debug state with an illegal value in DSPSR	r0p0, r1p0, r1p1	Open
<a href="#">2816903</a>	Programmer	Category C	PE might fail to detect multiple uncorrectable ECC errors in the L1 data cache tag RAM	r0p0, r1p0, r1p1	Open
<a href="#">3049877</a>	Programmer	Category C	L1D_TLB PMU event increments more than once per memory access	r0p0, r1p0	r1p1
<a href="#">3605050</a>	Programmer	Category C	Incorrect count for PMU event 0x004C (L1D_TLB_REFILL_RD) might be observed	r0p0, r1p0, r1p1	Open
<a href="#">3607348</a>	Programmer	Category C	PSTATE.{PAN,UAO} synchronization might not be honored while MSR PSTATE is speculative	r0p0, r1p0, r1p1	Open
<a href="#">3633467</a>	Programmer	Category C	EDSCR.STATUS not updated on Halting Step when a Load-Exclusive instruction generates a synchronous exception	r0p0, r1p0, r1p1	Open
<a href="#">3700181</a>	Programmer	Category C	PE might fail to log a RAS error for L2 data RAM ECC errors	r0p0, r1p0, r1p1	Open
<a href="#">3705917</a>	Programmer	Category C	PMU events are mis-categorized by not considering the effect of "Taken locally"	r0p0, r1p0, r1p1	Open



# Errata descriptions

## Category A

There are no errata in this category.

## Category A (rare)

### 1316063

### 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 and r1p0. Fixed in r1p1.

#### 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:
  - a. No load younger than ST1 has executed out-of-order for any of the following instructions:
    - i. Load.
    - ii. DMB.
    - iii. DSB.
    - iv. Atomic instruction which updates a register and has acquire semantics.
  - b. 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 CPUACTLR2\_EL1[16] to 1, hence preventing LD1 from speculating past ST1. This will have a performance impact on general workloads.

## Category B

### 1160841

### 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. Fixed in r1p0.

#### 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

Set CPUACTLR2\_EL1[0] to 1 and CPUACTLR2\_EL1[15] to 1.

## 1177367

### 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. Fixed in r1p0.

#### 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.

## 1191167

### 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. Fixed in r1p0.

#### 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 ELO.
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, ELOVTEN, ELOVCTEN, ELOPCTEN} to 0. If HCR\_EL2.{E2H,TGE}={1,1} then set CNTHCTL\_EL2.{ELOPTEN, ELOVTEN, ELOVCTEN, ELOPCTEN} to 0. The following registers will be trapped: CNTP\_CTL, CNTP\_CVAL, CNTP\_TVAL, CNTV\_CTL, CNTV\_CVAL, CNTV\_TVAL, CNTPCT, CNTVCT, CNTFRQ.

## 1204882

### 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. Fixed in r1p0.

#### 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 if software sets CPUACTLR2\_EL1 bit[11] to 0b1.



## 1220737

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

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0. Fixed in r1p0.

#### 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 CPUECTLR\_EL1[25:24] to 0b11, which disables write streaming to the L2. This will have an impact on performance for streaming workloads.

## 1253791

### 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. Fixed in r1p0.

#### 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.

## 1262841

### 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. Fixed in r1p0.

#### Description

Under specific conditions, an incorrect virtual to physical mapping might happen because the L2 TLB has been corrupted.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Since the last TLBI-ALLE1, TLB entries have been created where the stage1 mapping is larger than the stage2 mapping.
2. The CPU issues or receives a by-VA TLB operation for a VMID that is not used by the current translation regime.
3. Micro-architectural conditions occur.

The instructions affected by condition #3 are: TLBI VAAE1, TLBI VAAE1IS, TLBI VAALE1 and TLBI VAALE1IS.

#### Implications

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

#### Workaround

The workaround is to ensure the L2 TLB only contains EL1 or EL0 records for the current VMID, and no EL1 or EL0 records when executing at EL2 or higher.

EL2 and EL3 should execute:

- TLBI ALLE1
- DSB SY

as the first instructions when taking an exception from EL1 or EL0.

Where EL2 or EL3 use AT instructions against the EL1 or EL0 regime to produce a physical address from a virtual address, this should be followed by the above TLBI sequence.

Because of erratum #1165522 **Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation** there is a small chance that a speculated AT instruction at EL2 or EL3 creates TLB entries that match condition #1. Arm does not believe this is likely to coincide with condition #3.

The previous workaround of issuing TLBI VMALLE1 on exiting a guest VM did not cover cases where stage2 was disabled at EL1 or EL0, or describe how early the instruction must be issued.

The original workaround of setting CPUECTLR\_EL1[51], which disables the MMU hardware prefetcher, will not resolve this issue due to a subsequent erratum #1523503.

## 1273521

### 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. Fixed in r1p0.

#### 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:
  - a. Address translation is disabled.
  - b. The second half of the T32 instruction can be decoded as 16-bit instruction updating R15 (PC).
  - c. 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 to increase the mispredict to fetch latency, which will have some impact on performance.

## 1450698

### Software Step might prevent interrupt recognition

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0. Fixed in r1p1.

#### 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 ELO 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.

## 1467687

### Branch prediction for an ERET cached in the instruction cache might cause a deadlock

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

When a branch predictor makes a prediction for an ERET instruction, the core might deadlock.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core executes a conditional branch instruction.
2. The branch predictor caches the branch in Condition 1.
3. The branch instruction is overwritten by an ERET instruction by a self-modifying code sequence.
4. The core caches the ERET instruction in the instruction cache, and later fetches the ERET instruction from the cache.
5. The branch predictor makes a prediction for the ERET based on the branch information cached at Condition 2.
6. The predicted target matches ELR[PSTATE.EL].

#### Implications

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

#### Workaround

This erratum can be avoided by preventing the caching of the ERET. This can be done through the following write sequence to several IMPLEMENTATION DEFINED registers:

```
LDR x0,=0x3
MSR S3_6_c15_c8_0,x0
LDR x0,=0xF3D08000
MSR S3_6_c15_c8_2,x0
LDR x0,=0xFFF0F0FF
MSR S3_6_c15_c8_3,x0
LDR x0,=0x80000002003FF
MSR S3_6_c15_c8_1,x0
ISB
```



## 1508412

### NC/Device Load and Store Exclusive or PAR-Read collision can cause deadlock

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

Under certain conditions, execution of either a load to device or non-cacheable memory, and either a store exclusive or register read of the Physical Address Register (PAR\_EL1) in close proximity might lead to a deadlock.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

##### Case A

In program order:

1. The core executes a store-exclusive or register read of PAR\_EL1.
2. The core executes any load with device or non-cacheable memory attributes.

##### Case B

In program order:

1. The core executes any load with device memory attributes.
2. The core executes a store-exclusive or register read of PAR\_EL1.

#### Implications

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

#### Workaround

The following workaround is to prevent an ELO attack of the device. There is no workaround that will prevent malicious code at EL1 or higher from being able to exploit this erratum.

Case A of erratum can be avoided with the following steps:

1. Modify the software running at EL1 and above to include a DMB SY before and after accessing PAR\_EL1. PAR\_EL1 is not accessible from EL0.
2. Use the following write sequence to several IMPLEMENTATION DEFINED registers to have the hardware insert a DMB SY after all load-exclusive and store-exclusive instructions. The code sequence applies to r1p0 hardware and should be applied early in the boot sequence prior to any of the possible errata conditions being met:

```
LDR x0,=0x0
MSR S3_6_c15_c8_0,x0
LDR x0,=0x00e8400000
MSR S3_6_c15_c8_2,x0
LDR x0,=0x00ff600000
MSR S3_6_c15_c8_3,x0
LDR x0,=0x00e8c00080
MSR S3_6_c15_c8_4,x0
LDR x0,=0x00ffe000c0
MSR S3_6_c15_c8_5,x0
LDR x0,=0x04004003FF
MSR S3_6_c15_c8_1,x0
ISB
```

Note that if this workaround needs to be implemented on rOp0 hardware, then the following code sequence should be used instead of the above:

```
LDR x0,=0x0
MSR S3_6_c15_c8_0,x0
LDR x0,=0x00e8400000
MSR S3_6_c15_c8_2,x0
LDR x0,=0x00ffe00000
MSR S3_6_c15_c8_3,x0
LDR x0,= 0x4004003FF
MSR S3_6_c15_c8_1,x0
LDR x0,=0x1
MSR S3_6_c15_c8_0,x0
LDR x0,=0x00e8c00040
MSR S3_6_c15_c8_2,x0
LDR x0,=0x00ffe00040
MSR S3_6_c15_c8_3,x0
LDR x0,= 0x4004003FF
MSR S3_6_c15_c8_1,x0
ISB
```

In addition to the previous steps 1 and 2, Case B of the erratum also requires:

3. Prevent EL0 code from accessing a location mapped with device memory attributes.

Adding DMB after load with device memory attributes for Case-B is not expected to be required as explicit accesses between ldxr/stxr is unrealistic.

One implication of this workaround is that minor performance degradation might be observed in code utilizing load-exclusive and store-exclusive instructions.

## 1515815

The core might execute multiple instructions before taking a software step exception or halt step exception when the executing instruction resides in the L0 Macro-op cache

### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0. Fixed in r1p1.

### Description

When the core executes an instruction during an active-not-pending state in a software step or halt step process, the core might execute multiple instructions before taking a software step exception or halt step exception.

### Configurations Affected

This erratum affects all configurations.

### Conditions

1. Software step or halt step is enabled in the AArch64 instruction state.
2. Instruction fetch hits in the L0 Macro-op cache.

### Implications

If the above conditions are met, then the core might execute multiple instructions before taking a software step exception or halt step exception.

### Workaround

Set CPUACTLR\_EL1[11] to one, which flushes the L0 Macro-op cache for all context synchronization events.

**1791578**

## Atomic Store instructions to shareable write-back memory might cause memory consistency failures

### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r1p1. Open.

## Description

Atomic Store instructions to shareable write-back memory that are performed as far atomics might cause memory consistency failures if the initiating PE has a shared copy of the cache line containing the addressed memory.

## Configurations affected

This erratum affects all configurations that have an interconnect capable of handling far atomic transactions indicated by the BROADCASTATOMIC pin being set to 1.

## Conditions

1. PEO executes Atomic Store instruction that hits in the L1 data cache and L2 cache in the Shared state.
2. PEO changes the L2 state to Invalid, sends an invalidating snoop to the L1 data cache, and issues a AtomicStore transaction on the CHI interconnect.
3. PEO invalidating snoop to the L1 data cache is delayed due to internal queueing.

## Implications

If the above conditions are met, PEO might not observe invalidating snoops caused by other PEs in the same coherency domain and thus might violate memory consistency for loads to the same cache line as the Atomic Store.

## Workaround

Set CPUACTLR2\_EL1[2] to force Atomic Store operations to write-back memory to be performed in the L1 data cache.

## 1800714

### A transient single-bit ECC error in the MMU TC RAM might lead to stale translation in the L2 TLB

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

Under certain conditions, a transient single-bit ECC error in the MMU TC RAM might prevent a TLB invalidate (TLBI) instruction from removing the entry. If the transient error is not detected for a subsequent miss request targeting the affected page, then the MMU might return a stale translation.

#### Configurations affected

All configurations are affected.

#### Conditions

All of the following conditions must be met:

1. Both stage 1 and stage 2 translations are enabled.
2. Stage 1 page or block size is larger than stage 2 page or block size.
3. MMU TC RAM entry has a transient single-bit ECC error.
4. TLBI targets the translation in the MMU TC RAM entry containing the single-bit ECC error.
5. The single-bit ECC error prevents the TLBI from removing the entry.
6. Transient single-bit ECC error goes away before a subsequent translation request matching the L2 TLB entry is issued.

#### Implications

If the above conditions are met, then the MMU might return stale translation for a subsequent access. The transient single-bit ECC error will be reported in `ERRORMISCO_EL1` register.

#### Workaround

This erratum can be avoided by setting `CPUECTLR_EL1[53]` to 1, which disables the allocation of splintered pages in the L2 TLB.

## 1852353

### Watchpoint exception on Ld/St does not report correct address in FAR or EDWAR

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

If a load or store crosses a cache line (cache line size = 64 bytes) and a watchpoint address targets a location in the upper cache line, the Fault Address Register (FAR) or the External Debug Watchpoint Address Register (EDWAR) (if set up for Debug Halt) will contain an incorrect address.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

Incorrect address in FAR or EDWAR appears when the:

1. Watchpoint targets a double word (or less or more) at cache line address B.
2. Load or store targets accesses two cache lines: lower cache line A and upper cache line B. The cache line size is 64 bytes.

#### Implications

FAR contains the target address of load or store.

EDWAR contains the target address of load or store if enabled for Debug Halt.

#### Workaround

There is no hardware workaround.

The following software workaround can be applied:

If the Fault Address Register (FAR) or External Debug Watchpoint Address Register (EDWAR) does not match a watchpoint, software can attempt to identify a relevant watchpoint:



a) For A DC ZVA whose address is not aligned to DCZID\_EL0.BS, by rounding the faulting address down to a cache line boundary (64 bytes) and attempting to match this against active watchpoints.

Note: Most software aligns addresses used by DC ZVA, and this case is expected to be rare in practice.

b) For all other loads and stores, by attempting to use the address of the next cache line boundary (64 bytes) and attempting to match this against active watchpoints.

## 1923201

### External debugger access to Debug registers might not work during Warm reset

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

During Warm reset, external debugger access for Debug registers might be ignored.

#### Configurations Affected

All configurations are affected.

#### Conditions

1. Warm reset is asserted.
2. External debugger access is initiated for one of following Debug registers:
  - a. DBGBCR<n>\_EL1 (n=0-5)
  - b. DBGBVR<n>\_EL1 (n=0-5)
  - c. EDECCR

#### Implications

If the above conditions are met, the core might ignore the access request. The read operation might return incorrect data. The write operation might not take effect and stale data might be retained.

#### Workaround

There is no workaround.

## 1925769

### Store operation that encounters multiple hits in the TLB can access regions of memory with attributes that could not be accessed at that Exception level or Security state

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

Under certain circumstances, a store operation that encounters multiple hits in the TLB can generate a prefetch request to regions of memory with attributes that could not be accessed at that Exception level or Security state.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. A store operation encounters multiple hits in the TLB due to inappropriate invalidation or misprogramming of a contiguous bit.
2. A read request is generated with a physical address and attributes that are an amalgamation of the multiple TLB entries that hit.

#### Implications

If the above conditions are met, a read request could be generated to regions of memory with attributes that could not be accessed at that Exception level or Security state. The memory location will not be updated.

#### Workaround

This erratum can be avoided by setting CPUECTLR\_EL1[8] to 1. There is a small performance cost (<0.5%) for setting this bit.

## 1946167

### Atomic instructions with acquire semantics might not be ordered with respect to older stores with release semantics

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

Under certain conditions, atomic instructions with acquire semantics might not be ordered with respect to older instructions with release semantics. The older instruction could either be a store or store atomic.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Load atomic, CAS, or SWP with acquire but no release semantics is executed.
2. There is an older instruction with release semantics and it could either be a store to non-WB memory or a store atomic instruction that is executed as a far atomic.

#### Implications

If the above conditions are met, a memory ordering violation might happen.

#### Workaround

This erratum can be avoided by inserting a DMB ST before acquire atomic instructions without release semantics. This can be implemented through execution of the following code at EL3 as soon as possible after boot:

```
LDR x0,=0x4
MSR S3_6_c15_c8_0,x0
LDR x0,= 0x10E3900002
MSR S3_6_c15_c8_2,x0
LDR x0,= 0x10FFF00083
MSR S3_6_c15_c8_3,x0
LDR x0,= 0x2001003FF
MSR S3_6_c15_c8_1,x0
```

```
LDR x0,=0x5
MSR S3_6_c15_c8_0,x0
LDR x0,= 0x10E3800082
MSR S3_6_c15_c8_2,x0
LDR x0,= 0x10FFF00083
MSR S3_6_c15_c8_3,x0
LDR x0,= 0x2001003FF
MSR S3_6_c15_c8_1,x0
```

```
LDR x0,=0x6
MSR S3_6_c15_c8_0,x0
LDR x0,= 0x10E3800200
MSR S3_6_c15_c8_2,x0
LDR x0,= 0x10FFF003E0
MSR S3_6_c15_c8_3,x0
LDR x0,= 0x2001003FF
MSR S3_6_c15_c8_1,x0
```

ISB

## 2356587

### Continuous failing STREX because of another PE executing prefetch for store behind consistently mispredicted branch

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

A PE executing a PLDW or PRFM PST instruction that lies on a mispredicted branch path might cause a second PE executing a store exclusive to the same cache line address to fail continuously.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. One PE is executing store exclusive.
2. A second PE has branches that are consistently mispredicted.
3. The second PE instruction stream contains a PLDW or PRFM PST instruction on the mispredicted path that accesses the same cache line address as the store exclusive executed by the first PE.
4. PLDW/PRFM PST causes an invalidation of the first PE's caches and a loss of the exclusive monitor.

#### Implications

If the above conditions are met, the store exclusive instruction might continuously fail.

#### Workaround

Set CPUACTLR2\_EL1[0] to 1 to force PLDW/PRFM ST to behave like PLD/PRFM LD and not cause invalidations to other PE caches. There might be a small performance degradation to this workaround for certain workloads that share data.

## 2743100

### The core might deadlock during powerdown sequence

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

While powering down the *Processing Element* (PE), a correctable L2 tag ECC error might cause a deadlock in the powerdown sequence.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

This erratum occurs under the following conditions:

1. Error detection and correction is enabled through ERXCTLR\_EL1.ED=1.
2. PE executes more than 24 writes to Device-nGnRnE or Device-nGnRE memory.
3. PE executes powerdown sequence as described in the *Technical Reference Manual* (TRM).

#### Implications

If the above conditions are met, the PE might deadlock during the hardware cache flush that automatically occurs as part of the powerdown sequence.

#### Workaround

Add a DSB instruction before the ISB of the powerdown code sequence specified in the TRM.

## 3324348

### MSR PSTATE.SSBS to 0 is not fully self-synchronizing

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

When PSTATE.SSBS is written to 0, the Arm Architecture specifies that side-effects are guaranteed to be visible to later instructions in the Execution stream. However, for a window of time during speculative execution of **MSR PSTATE.SSBS**, speculative store data bypassing might still occur.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

The erratum occurs if the following condition applies:

**MSR PSTATE.SSBS** executes, setting PSTATE.SSBS to 0.

#### Implications

Security sensitive code executed shortly after **MSR PSTATE.SSBS** to 0 might not be fully protected by the *Speculative Store Bypass Safe* (SSBS) feature.

#### Workaround

Software at EL3, EL2, and EL1 should follow writes to the SSBS register with an *Instruction Synchronization Barrier* (ISB) instruction to ensure that the new value of PSTATE.SSBS affects subsequent instructions in the execution stream under speculation.

A kernel at EL1 or EL2 should not advertise the presence of MRS/MSR instructions to read/write the SSBS register from ELO. Arm expects that kernels provide system calls for ELO software to modify PSTATE.SSBS when the SSBS register is not implemented and that ELO software will use this when the presence of the SSBS register is not advertised.



## 3439000

### When Hardware Page Aggregation (HPA) is enabled memory accesses may be translated incorrectly

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0 and r1p1. Open.

#### Description

When Hardware Page Aggregation (HPA) is enabled memory accesses may be translated incorrectly. This may permit bypass of Stage-2 translation.

This issue has been assigned CVE ID CVE-2024-5660.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

The erratum occurs if all the following conditions apply:

1. Hardware page aggregation is enabled (CPUECTLR\_EL1[46]==0, which is the default value).
2. Stage-1 and/or Stage-2 translation is enabled for the active translation regime.
3. At Stage-1 or Stage-2 any of the following occur:
  - Translation table entries are modified to change the table or block size without following a Break-Before-Make approach.
  - Translation table entries within the same contiguous region have inconsistent values for the contiguous bit.
4. The translation table entries in condition 3 have inconsistent values for output addresses, access permissions, and/or memory attributes.
5. Complex, but not rare, microarchitectural conditions occur.

#### Implications

When all of the conditions above are met, any memory access translated by the translation table entries in condition 3 might use a Physical Address Space (PAS), Physical Address (PA), access permissions, and/or memory attributes which are not consistent with the architectural combination of Stage-1 translation and Stage-2 translation. Specifically any of the following may occur:

- The resulting PAS may be any arbitrary PAS reachable from the security state the access originated from:
  - For accesses originating from Non-secure state: Non-secure PAS only.
  - For accesses originating from Secure state: Secure or Non-secure PAS only.
- The resulting PA can be any arbitrary PA.
- The resulting access permissions can be any arbitrary access permissions.
- The resulting memory attributes can be any arbitrary memory attributes.

The resulting translation may permit software to read or write to an arbitrary PA which should not be accessible due to Stage-2 translation and/or may permit resulting memory attributes which should not be possible due to Stage-2 translation. Consequently this may allow software within a virtual machine to escalate privilege to EL2.

The resulting translation does not permit software in Normal state to read or write to any PA in the Secure PAS and consequently this does not provide a mechanism for software in Normal state to escalate privilege to Secure state.

## Workaround

The erratum can be avoided by setting CPUECTLR\_EL1[46] to 1, which will disable hardware page aggregation.

## 3696294

### Changing block size without break-before-make or mis-programming contiguous hint bit can lead to a livelock

#### Status

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r1p0 and r1p1. Open.

#### Description

Under certain conditions, changing block size without break-before-make or mis-programming the contiguous bit can lead to an interruptible livelock in violation of FEAT\_BBM level 2 requirements until TLB maintenance is performed.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

1. The contiguous bit is mis-programmed for a set of contiguous Stage-1 or Stage-2 translation table entries.
2. A load or store crosses a page boundary within a contiguous address range such that an access for one page is translated by a translation table entry with the contiguous bit set and an access for another page is translated via a translation table entry with the contiguous bit clear.

or

1. A Stage-1 or Stage-2 translation table entry is modified without break-before-make such that a VA or IPA which was previously translated by a Page or Block entry is subsequently translated via a larger Block entry.
2. No TLB maintenance is performed to remove TLB entries for the stale Page or Block entry.
3. A load or store crosses a page boundary such that accesses for either page could be translated via the new block entry, and at least one access could have been translated by a distinct Page or Block entry prior to modification.

#### Implications

When the previous conditions are met, the load or store instruction will stall indefinitely without raising a fault. During the stall, the load or stall can be interrupted.

#### Workaround

Where software which manages the translation tables cannot ensure that it is not subject to the stall conditions, or where stalling is unacceptable, software which manages the translation tables should ignore **ID\_AA64MMFR2\_EL1.BBM** and always follow a break-before-make approach.

Where software which manages the translation tables can ensure that it is not subject to the stall conditions, and it is acceptable to transiently stall lower privileged software, software which manages the translation tables should minimize the period for which the contiguous bit is mis-programmed and minimize the period between modifying a translation table entry and invalidating TLB entries for the previous translation table entry.

## Category B (rare)

1286809

### 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. Fixed in r1p1.

#### 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.

Note: For code sequences which have multiple TLB invalidate instructions followed by a single DSB, only the last TLB invalidate and DSB need to be repeated a second time.

## 1418842

### 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 and r1p0. Fixed in r1p1.

#### 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.{ELOPTEN, ELOVTEN, ELOVCTEN, ELOPCTEN} to 0. If HCR\_EL2.{E2H,TGE}={1,1} then set CNTHCTL\_EL2.{ELOPTEN, ELOVTEN, ELOVCTEN, ELOPCTEN} to 0. The following registers will be trapped:

- CNTP\_CTL.
- CNTP\_CVAL.
- CNTP\_TVAL.
- CNTV\_CTL.
- CNTV\_CVAL.
- CNTV\_TVAL.

- CNTPCT.
- CNTVCT.
- CNTFRQ.



## 1542418

### The core might fetch a stale instruction from the L0 Macro-op cache which violates the ordering of instruction fetches

#### Status

Fault Type: Programmer Category B Rare

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

When the core executes a direct branch that has been recently modified, associated with prefetch speculation protection, the core might fetch a stale instructions from the L0 Macro-op cache which violates the ordering of instruction fetches.

#### Configurations Affected

This erratum affects all multi-core configurations.

#### Conditions

1. The core is in AArch64 mode.
2. The modifying core changes instructions at address A.
3. The modifying core executes cache maintenance and synchronization instructions to make address A visible to all cores in the inner shareable domain.
4. A direct branch or a NOP is substituted with a direct branch targeting address A on the modifying core.
5. The executing core fetches the branch and correctly predicts the destination of the direct branch based on stale history due to ASID or VMID reuse.
6. Stale instructions are fetched from the L0 Macro-op cache, on the executing core, instead of the modified instructions at address A.

#### Implications

Software relying on prefetch speculation protection, instead of explicit synchronization when modifying a direct branch, might execute stale instructions when the branch is taken.

#### Workaround

This erratum can be avoided by invalidating branch history before reusing any ASID for a new address space. This can be done by ensuring 60 ASIDs are selected before any ASID is reused.

## Category C

### 1148171

### ERROMISCO might report incorrect BANK and SUBBANK values for transient parity errors in L1 instruction cache data array

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r1p0.

#### 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 ERROMISCO 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 ERROMISCO register might have incorrect information. This does not impact other fields in the ERROMISCO register that apply to the L1 instruction cache.

#### Workaround

There is no workaround for this erratum.

## 1151664

### Direct access to internal memory for L2 TLB might not update IDATAn\_EL3 registers

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r1p0.

#### Description

Direct access to internal memory for the L2 TLB might not update the IDATAn\_EL3 registers.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The RAMINDEX register is updated to initiate a direct access to the L2 TLB.
2. The instruction fetch unit is not processing any snoop requests.

#### Implications

Direct access to internal memory is a debug feature for reading contents of certain internal memories through IMPLEMENTATION DEFINED system registers. If the above conditions are met, then any direct access to the L2 TLB memory returns invalid data. The regional clock gating prevents update of the IDATAn\_EL3 register in the scenario.

#### Workaround

This erratum can be avoided by setting CPUACTLR\_EL1[14] to 1 for performing direct access to internal memory.

## 1162083

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

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r1p0.

#### 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.

## 1185469

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

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r1p0.

#### 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. TRCCONFIGR.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 ELO, then no workaround is required.

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

## 1192280

### 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. Fixed in r1p0.

#### 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.

## 1207839

### 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. Fixed in r1p0.

#### 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.

## 1220404

### 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. Fixed in r1p0.

#### 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.



## 1220843

### ERROSTATUS.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. Fixed in r1p0.

#### Description

The ERROSTATUS.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 ERROSTATUS.SERR = 0x12. Because of this erratum, they incorrectly record 0x18. Underrable data errors received from the interconnect should record ERROSTATUS.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 ERROCTLR.ED is 0.

#### Implications

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

#### Workaround

There is no workaround for this erratum.

## 1244986

### Illegal return event might corrupt PSTATE.UAO

#### Status

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

#### 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.

## 1256789

### 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. Fixed in r1p0.

#### 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\_ELO and DSPSR\_ELO 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.

## 1262908

### 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. Fixed in r1p0.

#### 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.

## 1328683

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

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### 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 error as an Unrecoverable (UEU) SError.

#### Workaround

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

## 1346768

### 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, and r1p1. 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.

## 1355135

### 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. Fixed in r1p1.

#### 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.

## 1395535

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

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### 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.



## 1405548

### MSR DSPSR\_ELO 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 and r1p0. Fixed in r1p1.

#### Description

An MSR DSPSR\_ELO 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\_ELO) 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.

## 1415321

### LDREX-STREX might succeed incorrectly when an intervening store occurs and LDREX detects a single-bit ECC error on the cache line in the L1 data cache tag RAM

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

If a core:

1. Detects a false miss due to a single-bit tag ECC error in the L1 data cache tag RAM on an LDREX.
2. Completes the LDREX by forwarding data from a prior store and that store is able to merge its data to the cache prior to a snoop targeting the same line, where the snoop is ordered ahead of the miss request from the load.

Then it might lead to the STREX succeeding even though there is an intervening store.

#### Configurations Affected

The erratum affects all multicore 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 an LDREX as part of a sequence to acquire a lock. The LDREX encounters a tag single-bit ECC error, which makes the line appear as a miss.
3. The LDREX allocates a miss request buffer, but is able to forward from the older store and complete. As a result the exclusive monitor is armed in Core A, and is tracking the outstanding miss request.
4. The older store drains to the cache as the line is still in the L1 data cache.
5. Core B sends a snoop for line X and the snoop is ordered ahead of the miss request from the load. Core A responds to the snoop, but the monitor is still armed, as it was tracking the outstanding miss request.
6. Core B performs a store to the line X.
7. Core A then receives the line X on behalf of its miss request and allocates the line.
8. STREX completes successfully as the monitor is armed.

#### Implications

If the above conditions are met, then the STREX can succeed, even though there was an intervening store in the middle of the LDREX-STREX sequence.

## Workaround

This erratum can be avoided by setting CPUACTLR3\_EL1[57].

## 1421023

### Portions of the branch target address recorded in ETM trace information are incorrect for an indirect branch with a malformed branch target address

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

The errant behavior described in this erratum pertains solely to ETM reporting information, and strictly to 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 fails to include the malformed target address information for the branch execution, but correctly includes this information when reporting exception information for the Instruction Abort. Only the upper and lower portions of the ETM branch target VA are erroneous, by nature of excluding the malformed address 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.

#### Implications

If the above conditions are met, the indirect branch with malformed target address will not include the malformed information in the branch target address 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 resultant Instruction Abort.

## 1487187

### 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. Fixed in r1p1.

#### 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.

## 1488613

### An unaligned load might initiate a prefetch request which crosses a page boundary

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

A load which crosses a 64-byte boundary, but not a 4KB boundary, and hits a TLB entry for a page which is less than 64KB in size, might trigger a prefetch request which incorrectly interprets the page size to be 64KB and therefore initiates a read request for an unexpected physical address.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The system is configured with read-sensitive Device memory at a physical address which overlaps with an aligned 64KB region that belongs to Normal memory.
2. A load which crosses a 64-byte boundary, but not a 4KB boundary, accesses the TLB in a one-cycle window and hits the entry which maps its virtual address, VA1, to physical address PA1.
3. The load triggers a prefetch request based on PA1 which might be outside of the page boundary for PA1, but within the 64KB aligned physical address region containing PA1.

#### Implications

If the above conditions are met, then the core might generate an unexpected read to a physical address within the 64KB aligned physical address region of the load.

#### Workaround

Arm does not expect read-sensitive Device memory to be mapped to a physical address which overlaps with a 64KB aligned physical address region belonging to Normal memory, therefore no workaround is necessary.

## 1491015

### TRCIDR3.CCITMIN value is incorrect

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0. Fixed in r1p1.

#### 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.

## 1514033

### Error Synchronization Barrier (ESB) instruction execution with a pending masked Virtual SError might not clear HCR\_EL2.VSE

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

If a Virtual SError is pending and masked at the current Exception level when an ESB instruction is executed, then the VDISR\_EL2 update occurs properly but in some cases the clearing of HCR\_EL2.VSE might not occur. This failure to clear HCR\_EL2.VSE can only occur when the Virtual SError is masked.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions:

1. A Virtual SError is pending at the current Exception level.
2. Virtual SErrors are masked at the current Exception level.
3. An ESB instruction executes.

#### Implications

If the above conditions are met, then under specific microarchitectural timing conditions HCR\_EL2.VSE might not be cleared to 0, which is required by the Arm architecture. This might result in spurious Virtual SErrors. Under all circumstances, the Virtual SError syndrome from VSESR\_EL2 is correctly recorded in VDISR\_EL2 and VDISR\_EL2.A is correctly set to 1.

#### Workaround

A workaround is not expected to be required. This is because existing software only executes ESB instructions at EL2 and above. If your software executes ESB instructions at EL1 with the conditions described above, then contact Arm support for more details.



## 1519163

### AMU Counter INST\_RETIRED does not increment correctly when 16 instructions retire in same cycle

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

Increments of count 16 results in an inaccurate accumulation of event INST\_RETIRED (event number 0x008), which counts instructions that are architecturally executed. All other counts less than 16 are correctly captured, and the counter is incremented properly.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The core enables the AMU.
2. The core enables counting on AMEVCNTR2\_ELO.
3. 16 instructions retire in the same cycle, comprising 8 fused instruction-pairs (where 2 instructions are fused into a single entity for processing in the CPU).

Reads of AMEVCNTR2\_ELO give inaccurate counts, as the counter does not increment when 16 instructions retire in the same cycle.

#### Implications

In the unlikely event of the erratum occurring, the inaccurate counts can give a lower than expected view of the instructions being retired on the core to any software profiling this activity.

#### Workaround

To workaround this issue, NOP elimination must be disabled. This is done by setting both CPUACTLR\_EL1[28] and CPUACTLR\_EL1[26] to 1'b1. The performance delta of this workaround is expected to be small.

## 1522097

The core might detect a breakpoint exception one instruction earlier than the programmed location when the L0 Macro-op cache contains an instruction that is affected by a parity error

### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

### Description

When an address matching breakpoint is set to the instruction following an instruction that is affected by a parity error, the core might detect a breakpoint exception on the instruction with the parity error.

### Configurations Affected

This erratum affects the configuration with `CORE_CACHE_PROTECTION = 1`.

### Conditions

1. The core is in AArch64 state.
2. An instruction that is cached in L0 Macro-op cache has a parity error.
3. An address matching breakpoint is marked on the instruction right after the above parity error instruction.

### Implications

If the above conditions are met, then the core might detect a breakpoint exception at the instruction with the parity error, which is incorrect.

### Workaround

This erratum has no workaround.

## 1523503

### CPUECTLR\_EL1 controls for the MMU have no affect

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

The CPUECTLR\_EL1 register contains IMPLEMENTATION DEFINED configuration and control options for the MMU. The MMU bits affected by this erratum are CPUECTLR\_EL1[54:46]. Any changes to these values have no affect on the functionality or performance.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions:

Software updates to modify MMU control bits CPUECTLR\_EL1[54:46] from reset values have no affect.

#### Implications

Software attempts to change the functionality or performance of the core by changing reset values of CPUECTLR\_EL1[54:46] have no affect. The value is updated in the register correctly, such that any subsequent read of the CPUECTLR\_EL1 register returns the expected data, however, the modifications have no affect on the behavior of the core.

#### Workaround

There is no workaround.

## 1610369

### ERR0MISCO\_EL1.SUBARRAY value for ECC errors in the L1 data cache might be incorrect

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0. and r1p1. Open.

#### Description

Under certain conditions, the ERR0MISCO\_EL1.SUBARRAY value recorded for ECC errors in the L1 data cache might be incorrect.

#### Configurations Affected

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

#### Conditions

1. A load, store, or atomic instruction accesses multiple banks of the L1 data cache.
2. One of the banks accessed has an ECC error.

#### Implications

If the above conditions are met, then ERR0MISCO\_EL1.SUBARRAY might have an incorrect value. The remaining fields of the ERR0MISCO\_EL1 register remain correct.

#### Workaround

There is no workaround for this erratum.

## 1624431

### CPUAMEVTYPER4\_ELO register cannot be written

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

#### Description

The AMU activity monitor counter 4 is documented as programmable at EL3, allowing users to select between two different event types. In order to select event type 0xF2, "Max Power Mitigation Mechanism", the appropriate event number must be selected by writing it to the CPUAMEVTYPER4\_ELO register. However, attempts to write the CPUAMEVTYPER4\_ELO register at EL3 result in an UNDEFINED trap.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. MSR instruction attempts to write CPUAMEVTYPER4\_ELO register at EL3.

#### Implications

The AMU activity counter 4 is only able to count the event 0xF1 ("High Activity").

#### Workaround

A workaround is not expected to be required. If you require access to the "Max Power Mitigation Mechanism" event type, then contact Arm support for more details.

## 1655748

### MRC read of DBGDSCRint into APSR\_nzcv might produce wrong results and lead to corruption

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

In AArch32, MRC reads of DBGDSCRint into destination APSR\_nzcv (Rt=15) always produce a result of 0. Also, if there is a younger MRC or MRRC read to any accessible register following the DBGDSCRint read into APSR\_nzcv, then the younger read result might be corrupted.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

1. The core is in AArch32 state at ELO.
2. An MRC read of DBGDSCRint into APSR\_nzcv (Rt=15) occurs.

#### Implications

If the above conditions are met, then:

1. APSR\_nzcv is always written with 0.
2. Under specific microarchitectural timing conditions in AArch32 ELO, a subsequent MRC or MRRC might be corrupted.

#### Workaround

Directly read DBGDSCRint with an MRC instruction into a general-purpose register (R0-R14), and then write that general-purpose register to the flags by doing an MSR APSR\_f. To avoid the possible corruption, add an ISB instruction before any subsequent MRC or MRRC instructions.

## 1662411

### Executing a cache maintenance by set/way instruction targeting the L1 data cache in the presence of snoops might result in a deadlock

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

Under certain conditions, executing a cache maintenance by set/way instruction targeting the L1 data cache in close proximity to multiple snoops where the older snoop detects a transient ECC error might result in a deadlock.

#### Configurations Affected

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

#### Conditions

1. The core has executed at least two snoop requests looking up the L1 data cache. These could have been generated internally from this core or from another core in the system.
2. The older snoop detects a transient single-bit or double-bit ECC error, but at least two snoops have performed a lookup of the L1 data cache.
3. The core executes a cache maintenance by set/way instruction targeting the L1 data cache.
4. The snoops are required to perform another lookup due to the ECC error detected. All snoops are rescheduled to maintain ordering of the snoop transactions.
5. The snoop transactions continuously retry the L1 data cache lookup, preventing the cache maintenance operation from completing.

#### Implications

If the above conditions are met under certain timing conditions, then the snoops might not make progress, resulting in a deadlock. Arm does not expect cache maintenance operations by set/way to be executed in most code sequences, since hardware mechanisms have been incorporated for flushing the caches as a part of powerdown sequences. Software is expected to use cache maintenance operations by VA to manage coherency.

Note that cache maintenance by set/way instructions are `UNDEFINED` at `ELO`.

#### Workaround

Software should avoid the use of cache maintenance operations by set/way. A hypervisor should trap these instructions by setting HCR\_EL2.TSW = 1 and emulate the instructions with equivalent cache maintenance operations by virtual address for the entire address space of the guest.



## 1702492

### The core might not update IDATA\*\_EL3 correctly by a direct memory access to L1 Instruction Cache Tag or L1 Instruction TLB

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

CPU might not update IDATA\*\_EL3 correctly when a direct memory access to L1 Instruction Cache Tag or L1 Instruction TLB is initiated.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

When one of the following conditions is satisfied, the CPU can produce this erratum:

1. A direct memory access to L1 Instruction Cache Tag is initiated while the core is processing IC IALLU or IC IALLUIS.
2. A direct memory access to L1 Instruction TLB is initiated while an address translation was disabled in EL3.

#### Implications

IDATA\*\_EL3 might not be updated after the completion of the direct memory access. IDATA\*\_EL3 might hold either an old value for L1 Instruction Cache Tag access or a corrupted value for L1 Instruction TLB access.

#### Workaround

This erratum has no workaround.

## 1788065

### Possible loss of CTI event

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

A CTI event from the core to the external DebugBlock might be dropped, in rare occurrences, if close in temporal proximity to a previous CTI event.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. CTI event occurs.
2. Another CTI event occurs before completion of the processing of the previous CTI event.

#### Implications

CTI events might be dropped.

#### Workaround

This erratum has no workaround.

## 1788067

### Loss of CTI events during warm reset

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

ETM external output CTI events from the core to the external DebugBlock will not be reported during warm reset.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. An ETM external output CTI event occurs while warm reset is asserted.

#### Implications

The ETM external output CTI event will be dropped and any cross triggering that depends on this CTI event will not occur. For example, if the ETM external output was to be used to trigger a trace capture component to stop trace capture, then trace capture will not stop due to this event.

#### Workaround

This erratum has no workaround.

## 1791402

### The core might deadlock when an external debugger injects instructions using ITR register

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

The core might deadlock when an external debugger injects instructions by ITR register.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

1. An external debugger requests the core to enter debug state while the core is stalled because of an instruction abort due to a permission fault.
2. The external debugger injects instructions using the ITR register.

#### Implications

The core might deadlock if the above conditions are satisfied.

#### Workaround

This erratum has no workaround.

## 1827134

### External debug accesses in memory access mode with SCTL<sub>R</sub>\_EL<sub>x</sub>.IESB set might result in unpredictable behavior

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open

#### Description

In Debug state with SCTL<sub>R</sub>\_EL<sub>x</sub>.IESB set to 1, memory uploads and downloads executed in memory access mode might lead to unpredictable behavior for the current exception level.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Core is In Debug state.
2. SCTL<sub>R</sub>\_EL<sub>x</sub>.IESB is set to 1 for the current exception level.
3. Memory access mode is enabled via EDSCR.MA set to 1.

#### Implications

If the above conditions are met, memory upload and download behavior is unpredictable for the current exception level and might lead to incorrect operation or results. The unpredictable behavior is limited to legal behavior at the current exception level.

#### Workaround

The erratum can be avoided by clearing SCTL<sub>R</sub>\_EL<sub>x</sub>.IESB before performing memory uploads or downloads in Debug state using memory access mode.

## 1830646

### Watchpoint Exception on DC ZVA does not report correct address in FAR or EDWAR

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

If the watchpoint address targets a lower portion of a cache line, but not all of the cache line, and the address target of the Data Cache Zero by VA (DC ZVA) falls in the upper portion of the cache line that the watchpoint does not target, the Fault Address Register (FAR) (or External Debug Watchpoint Address Register (EDWAR) if setup for Debug Halt) will contain an incorrect address.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Watchpoint targets double word (or less or more) at address A.
2. DC ZVA targets address greater than A+7, but less than A+63. The cache line size is 64 bytes, which is a mis-aligned address.

#### Implications:

FAR contains target address of DC ZVA.

EDWAR contains target address of DC ZVA if enabled for Debug Halt.

#### Workaround:

There is no hardware workaround. The common case for DC ZVA targets is to be granule aligned, thus most software will not be affected by this case.

## 1857204

**A memory mapped write to PMSSRR might falsely cause some PMU counters and counter overflow status to be reset after snapshot capture and read might return unknown/written data**

### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

### Description:

A memory mapped write to PMSSRR at offset 0x6f4 might configure the Cycle counter and/or Performance Monitor event counters to be reset along with reset of corresponding overflow status bits in the PMOVSRR register. The register supports read/write functionality instead of RAZ/WI.

### Configurations affected

This erratum affects all configurations.

### Conditions

1. System enables PMU snapshot mechanism.
2. System performs memory mapped write of PMSSRR setting PMSSRR[x], where x is 31 or any value from 0 to 5 (inclusive).
3. Snapshot trigger is seen through a legal mechanism.

### Implications

If the above conditions are met, the corresponding counter (PMCCNTR\_ELO if x=31 or PMEVCNTR<x>\_ELO if x = [0,5]) will reset after a snapshot is taken. Further, the corresponding bit in the PMOVSRR\_ELO register will be reset.

A memory mapped read will return data that is written to these bits and 0 otherwise.

This register is supposed to have RAZ/WI functionality and no effect on other counters.

### Workaround

Avoid write of PMSSRR when system is using the PMU Snapshot mechanism.

## 1857621

### Uncorrectable tag errors in L2 cache might cause deadlock

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

Under rare conditions that include the aliasing of multiple virtual addresses to a single physical address, a detected and reported double-bit ECC error in the L2 cache tag RAM might lead to a state in which an unexpected L1 cache eviction can cause a deadlock in the L2 cache.

#### Configurations affected

This erratum affects configurations with `CORE_CACHE_PROTECTION TRUE`.

#### Conditions

1. L2 cache detects and reports a tag double-bit ECC error.
2. A set of rare conditions occur within the PE memory system.

#### Implications

If the above conditions are met, the L2 transaction queue might deadlock and never complete the prefetch operation.

#### Workaround

There is no workaround for this erratum.



**1869877****ERR0MISCO\_EL1.SUBARRAY, ERROSTATUS.CE and ERROSTATUS.DE values for ECC errors in the L1 data cache might be incorrect****Status**

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

**Description**

Under certain conditions, the ERR0MISCO\_EL1.SUBARRAY, ERROSTATUS.CE and ERROSTATUS.DE values recorded for ECC errors in the L1 data cache might be incorrect.

**Configurations affected**

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

**Conditions**

1. The L1 data cache contains both a single-bit and double-bit ECC error on different words within the same 64-byte cacheline.
2. An access is made to the cacheline in the L1 data cache containing both the single-bit and double-bit ECC errors simultaneously.

**Implications**

If the above conditions are met, then ERR0MISCO\_EL1.SUBARRAY, ERROSTATUS.CE and ERROSTATUS.DE might have an incorrect values.

**Workaround**

There is no workaround for this erratum.

## 1880115

### Noncompliance with prioritization of Exception Catch debug events

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

ARMv8.2 architecture requires that Debug state entry due to an Exception Catch debug event (generated on exception entry) occur before any asynchronous exception is taken at the first instruction in the exception handler. An asynchronous exception might be taken as a higher priority exception than Exception Catch and the Exception Catch might be missed altogether.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Debug Halting is allowed.
2. EDECCR bits are configured to catch exception entry to ELx.
3. A first exception is taken resulting in entry to ELx.
4. A second, asynchronous exception becomes visible at the same time as exception entry to ELx.
5. The second, asynchronous exception targets an Exception level ELy that is higher than ELx.

#### Implications

If the above conditions are met, the core might recognize the second exception and not enter Debug state as a result of Exception Catch on the first exception. When the handler for the second exception completes, software might return to execute the first exception handler, and assuming the core does not halt for any other reason, the first exception handler will be executed and entry to Debug state via Exception Catch will not occur.

#### Workaround

When setting Exception Catch on exceptions taken to an Exception level ELx, the debugger should do either or both of the following:

1. Ensure that Exception Catch is also set for exceptions taken to all higher Exception Levels, so that the second (asynchronous) exception generates an Exception Catch debug event.
2. Set Exception Catch for an Exception Return to ELx, so that when the second (asynchronous)

exception handler completes, the exception return to ELx generates an Exception Catch debug event.

Additionally, when a debugger detects that the core has halted on an Exception Catch to an Exception level ELy, where  $y > x$ , it should check the ELR\_ELy and SPSR\_ELy values to determine whether the exception was taken on an ELx exception vector address, meaning an Exception Catch on entry to ELx has been missed.

## 1884878

The core might report incorrect fetch address to FAR\_ELx when the core is fetching an instruction from a virtual address associated with a page table entry which has been modified

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

### Description

When a core fetches an instruction from a virtual address that is associated with a page table entry which has been modified and the fetched block is affected by parity error, the core might report an incorrect address within the same 32B block onto the Fault Address Register (FAR).

### Configurations Affected

All configurations are affected.

### Conditions

1. The core fetches instructions from an aligned 32B virtual address block.
2. A page table entry associated with the above 32B aligned block is updated. The new translation would cause an instruction abort.
3. TLB holds the old translation since the synchronization process, for example, TLB Invalidate (TLBI) followed by Data Synchronization Barrier (DSB), was not completed.
4. Some of the fetched instructions are affected by parity error in I-cache data RAM.
5. Context synchronization events were not processed between the last executed instruction and the above instruction.

### Implications

When the above conditions are satisfied, a core might report an incorrect fetch address to FAR\_ELx. The address reported in FAR\_ELx points at an earlier location in the same aligned 32B block. FAR\_ELx[63:5] still points correct virtual address.

### Workaround

There is no workaround.

## 1885949

### L2 data RAM may fail to report corrected ECC errors

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

For specific operation types and cache states, a read of the L2 data RAM might fail to report a detected and corrected single-bit ECC error.

#### Configurations affected

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

#### Conditions

1. PE L1 data cache and L2 cache are in a SharedClean state and the exclusive monitor is armed for a given physical address.
2. PE executes a store exclusive instruction to this physical address.
3. L2 cache reads its data RAMs, and detects and corrects a single-bit ECC error.

#### Implications

If the above conditions are met, the PE will correct the error, but might fail to report it in the RAS error log registers. This can cause a small loss in diagnostic capability.

#### Workaround

There is no workaround.

## 1899210

### Some corrected errors might incorrectly increment ERR0MISC0.CECR or ERR0MISC0.CECO

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

If a Corrected Error is recorded because of a bus error which has no valid location (ERR0STATUS.MV=0x0), then a subsequent Corrected Error might incorrectly increment either of the ERR0MISC0.CECR or ERR0MISC0.CECO counters.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. A Corrected Error which has no valid location (ERR0STATUS.MV=0x0) is recorded.
2. A subsequent Corrected Error occurs.

#### Implications

The subsequent Corrected Error might improperly increment either of the ERR0MISC0.CECR or ERR0MISC0.CECO counters.

#### Workaround

No workaround is expected to be required.

## 1899434

### PFG duplicate reported faults through a Warm reset

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open

#### Description

Under certain conditions, the Pseudo-fault Generation Error Record Registers might generate duplicate faults through a Warm reset.

#### Configurations affected

All configurations are affected.

#### Conditions

1. ERROPFGCDN is set with a non-zero countdown value.
2. ERROPFGCTL is set to generate a pseudo-fault with ERROPFGCTL.CDEN enabled.
3. The countdown value expires, generating a pseudo-fault.
4. Warm reset asserts.

#### Implications

After the Warm reset, a second generated pseudo-fault might occur.

#### Workaround

De-assert the ERROPFGCTL control bits before asserting a Warm reset.

## 1923198

### IDATAn\_EL3 might represent incorrect value after direct memory access to internal memory for Instruction TLB

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

After implementation-defined RAMINDEX register is programmed to initiate direct memory access to internal memory for Instruction TLB, implementation-defined IDATAn\_EL3 value represents unpredictable value.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. Implementation-defined RAMINDEX register is programmed to initiate direct memory access to internal memory for Instruction TLB.

#### Implications

If the above conditions are met, IDATAn\_EL3 register might represent incorrect value for Translation regime, VMID, ASID, and VA[48:21].

#### Workaround

There is no workaround.



## 1950447

### The PE might deadlock if Pseudofault Injection is enabled in Debug State

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

If Pseudofault Injection is enabled for the PE node (ERR0PFGCTL.CDNEN=0x1) and the PE subsequently enters Debug State, then the PE might deadlock. Alternatively, if the PE is executing in Debug State and the PE enables Pseudofault Injection for the PE node (ERR0PFGCTL.CDNEN=0x1), then the PE might deadlock.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. ERR0PFGCTL.CDNEN is set to 0x1 to enable Pseudofault Injection.
2. The PE enters Debug State.

OR

1. The PE is executing in Debug State.
2. ERR0PFGCTL.CDNEN is set to 0x1 to enable Pseudofault Injection.

#### Implications

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

#### Workaround

Ensure ERR0PFGCTL.CDNEN=0x0 before entering Debug State and while executing in Debug State.

## 2001419

### DRPS might not execute correctly in Debug state with SCTLR\_ELx.IESB set in the current EL

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

In Debug state with SCTLR\_ELx.IESB set to 1, the **DRPS** (debug only) instruction does not execute properly. Only partial functionality of the **DRPS** instruction is performed.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

The erratum occurs under the following conditions:

1. The core is in Debug state.
2. SCTLR\_ELx.IESB is set to 1 for the current Exception level.
3. A **DRPS** instruction is executed.

#### Implications

If the above conditions are met, the **DRPS** instruction does not complete as intended, which might lead to incorrect operation or results. Register data or memory will not be corrupted. There are also no security or privilege violations.

#### Workaround

The erratum can be avoided by clearing SCTLR\_ELx.IESB followed by the insertion of an **ISB** and an **ESB** instruction in the code before the DRPS instruction.

## 2033523

### ETM trace information records a branch to the next instruction as an N atom

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

If a branch is taken to the next instruction, and if the instruction state remains the same, then the ETM traces it as an N atom rather than an E atom or branch address packet. This is incorrect as the ETM architecture says a taken branch should be traced as an E atom. This affects all forms of branches. State-changing branches are traced correctly.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

This issue might occur when:

1. ETM is enabled.
2. A branch is taken to the next instruction.
3. The instruction state does not change.

#### Implications

A trace decoder that interprets an N atom to move to the next instruction in the same state without a push or pop from the return stack will correctly maintain the control flow but will not be able to infer anything from a conditional branch.

A trace decoder that checks if unconditional branches were not traced as N atom might report an error.

#### Workaround

To ensure continued control flow, ensure the trace decoder always interprets an N atom to move to the next instruction in same state without a push or pop from the return stack.

## 2052427

### An execution of MSR instruction might not update the destination register correctly when an external debugger initiates an APB write operation to update debug registers

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

When an **MSR** instruction and an APB write operation are processed on the same cycle, the **MSR** instruction might not update the destination register correctly.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

This erratum occurs under the following conditions:

1. A CPU executes an **MSR** instruction to update any of following SPR registers:
  - a. DBGBCR<n>\_EL1
  - b. DBGBVR<n>\_EL1
  - c. DBGWCR<n>\_EL1
  - d. DBGWVR<n>\_EL1
  - e. OSECCR\_EL1
2. An external debugger initiates an APB write operation for any of following registers:
  - a. DBGBCR<n>
  - b. DBGBVR<n>
  - c. DBGBXVR<n>
  - d. DBGWCR<n>
  - e. DBGWVR<n>
  - f. DBGWXVR<n>
  - g. EDECCR
  - h. EDITR
3. The SPR registers (for example, OSLSR\_EL1.OSLK and EDSCR.TDA) and external pins are programmed to allow the following behavior:
  - a. The execution of an **MSR** instruction in condition 1 to update its destination register without neither a system trap nor a debug halt
  - b. The APB write operation in condition 2 to update its destination register without error
4. The **MSR** instruction execution in condition 1 and APB write operation in condition 2 happen in same

cycle.

5. The **MSR** write and the APB write are to two different registers. The architecture specifies that it is the software or debugger's responsibility to ensure writes to the same register are updated as expected.

## Implications

If the above conditions are met, an execution of the **MSR** instruction might not update the destination register correctly. The destination register might contain one of following values after execution:

1. The execution of the **MSR** instruction is ignored. The destination register of the **MSR** instruction holds an old value.
2. The execution of the **MSR** instruction writes an incorrect value to its destination register.

A external debugger and system software are expected to be coordinated to prevent conflict in these registers.

## Workaround

No workaround is required for this erratum.

## 2110734

### External APB write to a register located at offset 0x084 might incorrectly issue a write to External Debug Instruction Transfer Register

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

The core might incorrectly issue a write to External Debug Instruction Transfer Register (EDITR) when an external APB write to another register that is located at offset 0x084 is performed in the Debug state. The following debug components share the offset alias with the EDITR register:

- ETE - TRCVIIECTLR - ViewInst Include/Exclude Control Register
- Reserved locations

The following debug component shares the offset alias with the EDITR register when the PE is configured with 20-PMUs:

- PMU - PMEVCNTR16[63:32] - Event Counter 16

#### Configurations affected

This erratum affects all configurations.

#### Conditions

1. The core is in debug state.
2. The External Debug Status and Control Register (EDSCR) cumulative error flag field is 0b0.
3. Memory access mode is disabled, in example, EDSCR.MA = 0b0.
4. The OS Lock is unlocked.
5. External APB write is performed to a memory mapped register at offset 0x084 other than the EDITR.

#### Implications

If the above conditions are met, then the core might issue a write to the EDITR and try to execute the instruction pointed to by the ITR. As a result of the execution, the following might happen:

- CPU state and/or memory might get corrupted.
- The CPU might generate an UNDEFINED exception.
- The EDSCR.ITE bit will be set to 0.

## Workaround

Before programming any register at this offset when the PE is in Debug state, the debugger should either:

- Set the EDSCR.ERR bit by executing some Undefined instruction (e.g. writing zero to EDITR); or
- Set the OS Lock and then unlock it afterwards.

## 2141649

### A64 WFI or A64 WFE executed in Debug state suspends execution indefinitely

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

Executing an A64 WFI or WFE instruction while in Debug state results in suspension of execution, and execution cannot be resumed by the normal WFI or WFE wake-up events while in Debug state.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The Processing Element (PE) is in Debug state and in AArch64 Execution state.
2. A WFI or WFE instruction is executed from EDITR.

#### Implications

If the above conditions are met, the PE will suspend execution.

This is not thought to be a serious erratum, because an attempt to execute a WFI or WFE instruction while in Debug state is not expected.

For WFI executed in Debug state, execution can only resume by any of the following:

- A Cold or Warm reset
- A Restart request trigger event from the Cross Trigger Interface (CTI) causing exit from Debug state, followed by a WFI wake-up event

For WFE executed in Debug state, execution can only resume by any of the following:

- A Cold or Warm reset
- A Restart request trigger event from the CTI causing exit from Debug state, followed by a WFE wake-up event
- An external event that sets the Event Register. Examples include executing an SEV instruction on another PE in the system or an event triggered by the Generic Timer.

#### Workaround



A workaround is unnecessary, because an attempt to execute a WFI or WFE instruction while in Debug state is not expected.

## 2238113

### Reads of DISR\_EL1 incorrectly return 0s while in Debug State

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

When the Processing Element (PE) is in Debug State, reads of DISR\_EL1 from EL1 or EL2 with SCR\_EL3.EA=0x1 will incorrectly return 0s.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The PE is executing in Debug State at EL1 or EL2, with SCR\_EL3.EA=0x1.
2. The PE executes an MRS to DISR\_EL1.

#### Implications

If the above conditions are met, then the read of DISR\_EL1 will incorrectly return 0s.

#### Workaround

No workaround is expected to be required.

## 2239142

### DRPS instruction is not treated as UNDEFINED at EL0 in Debug state

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

In Debug state, DRPS is not treated as an UNDEFINED instruction.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

1. The Processing Element (PE) is in Debug state.
2. PE is executing at EL0.
3. PE executes DRPS instruction.

#### Implications

If the above conditions are met, then the PE will incorrectly execute DRPS as NOP instead of treating it as an UNDEFINED instruction.

#### Workaround

There is no workaround.

## 2263696

### L1 Data poison is not cleared by a store

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

The L1 Data poison is not cleared by a store under certain conditions.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

This erratum occurs under the following conditions:

1. A Processing Element (PE) executes a store that does not write a full word to a location that has data marked as poison.
2. The PE executes another store that writes to all bytes that contain data poison before the previous store is globally observable.

#### Implications

If the above conditions are met, then the poison bit in the L1 Data cache does not get cleared.

#### Workaround

This erratum can be avoided by inserting a DMB before and after a word-aligned store that is intended to clear the poison bit.

## 2278130

### PMU L1D\_CACHE\_REFILL\_OUTER is inaccurate

#### Status

Fault Type: Programmer Category C  
Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

The L1D\_CACHE\_REFILL\_OUTER PMU event 0x45 is inaccurate due to ignoring refills generated from a system cache. The L1D\_CACHE\_REFILL PMU event 0x3 should be the sum of PMU events L1D\_CACHE\_REFILL\_INNER 0x44 and L1D\_CACHE\_REFILL\_OUTER 0x45, however, due to the inaccuracy of L1D\_CACHE\_REFILL\_OUTER 0x45 it is possible that this might not be the case.

Note: L1D\_CACHE\_REFILL PMU event 0x3 does accurately count all L1D cache refills, including refills from a system cache.

#### Configurations Affected

This erratum affects all configurations which implement a system cache.

#### Conditions

This erratum occurs under the following conditions:

1. The L2 inner cache is allocated with data transferred from a system cache.

#### Implications

When the previous condition is met, the L1D\_CACHE\_REFILL\_OUTER PMU event 0x45 does not increment properly.

#### Workaround

The correct value of L1D\_CACHE\_REFILL\_OUTER PMU event 0x45 can be calculated by subtracting the value of L1D\_CACHE\_REFILL\_INNER PMU event 0x44 from L1D\_CACHE\_REFILL PMU event 0x3.

## 2307836

### ESR\_ELx.ISV can be set incorrectly for an external abort on translation table walk

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

When a data double bit error or external abort is encountered during a translation table walk, a synchronous exception is reported with the ISV bit set in the ESR\_ELx register.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

This erratum occurs under the following condition:

1. A data double bit error or external abort is encountered during a translation table walk, and a synchronous exception is reported.

#### Implications

If the previous condition is met, the ESR\_ELx.ISV bit will be set. The ESR[23:14] bits are set with the correct syndrome for the instruction making the access. That is SAS, SSE, SRT, SF, and AR are all set according to the instruction.

#### Workaround

This erratum has no workaround.

## 2391682

### Software-step not done after exit from Debug state with an illegal value in DSPSR

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, r1p1. Open.

#### Description

On exit from Debug state, PSTATE.SS is set according to DSPSR.SS and DSPSR.M.

If DSPSR.M encodes an illegal value, then PSTATE.SS should be set according to the current Exception level. When the erratum occurs, the PE always writes PSTATE.SS to 0.

#### Configurations Affected

This erratum affects all configurations.

#### Conditions

- Software-step is enabled in current Exception level
- DSPSR.M encodes an illegal value, like:
  - M[4] set
  - M is a higher Exception level than current Exception level
  - M targets EL2 or EL1, when they are not available
- DSPSR.D is not set
- DSPSR.SS is set

#### Implications

If the previous conditions are met, then, on exit from Debug state the PE will directly take a Software-step Exception, without stepping an instruction as expected from DSPSR.SS=1.

#### Workaround

This erratum has no workaround.

## 2816903

### PE might fail to detect multiple uncorrectable ECC errors in the L1 data cache tag RAM

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0, and r1p1. Open.

#### Description

Under certain conditions, the *Processing Element* (PE) might fail to report multiple uncorrectable *Error Correction Code* (ECC) errors that occur in the L1 data cache tag RAM.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

1. The PE detects and reports an uncorrectable ECC error in the L1 data cache tag RAM.
2. The PE detects a second uncorrectable ECC error in the L1 data cache tag RAM and an uncorrectable ECC error in the L1 data cache data RAM.

#### Implications

If the previous conditions are met, then the PE might fail to report the second uncorrectable ECC error in the L1 data cache tag RAM and the address recorded in `ERR0ADDR` might have an incorrect value. The ECC error occurring in the L1 data cache data RAM is reported correctly.

#### Workaround

No workaround is necessary. This erratum represents a condition where multiple uncorrectable ECC errors occur in a short period of time. While the PE does not report the errors correctly, ECC still provides a valuable mechanism for error detection and correction.



**3049877****L1D\_TLB access related PMU event increments more than once per memory access****Status**

Fault Type: Programmer Category C

Fault Status: Present in r0p0 and r1p0. Fixed in r1p1.

**Description**

The L1D\_TLB access related PMU event 0x25 increments more than once per memory access due to TLB miss and refill conditions. This might lead to inconsistencies between other TLB related events, such as, L1D\_TLB\_REFILL PMU event 0x5 or attributable L1 data TLB miss rate.

**Configurations affected**

This erratum affects all configurations.

**Conditions**

This erratum occurs under the following conditions:

1. MMU is enabled.
2. Memory accesses result in a significant number of misses to the L1 data TLB.

**Implications**

Memory accesses which result in a significant number of L1 data TLB misses might increment L1D\_TLB PMU event 0x25 more than expected exposing inconsistencies with other related L1 data TLB events.

**Workaround**

There is no workaround for this erratum.

## 3605050

### Incorrect count for PMU event 0x004C (L1D\_TLB\_REFILL\_RD) might be observed

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0 and r1p1. Open.

#### Description

A hardware generated prefetch operation or a PRFM instruction might indicate a L1D\_TLB\_REFILL\_RD event leading to an incorrect count.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

The erratum occurs if all the following conditions apply:

1. PMU counters are configured to count event 0x004C.
2. A hardware generated prefetch or PRFM instruction might encounter a L1D TLB miss, resulting in a refill operation and triggering event 0x004C.

#### Implications

If the previous conditions are met, the count indicated by event 0x004C will not reflect the conditions specified in the Arm Architecture Reference Manual. Furthermore, this event is used in calculating the "Attributable Level 1 TLB refill rate, read" metric which by extension will not reflect an accurate rate.

#### Workaround

No workaround is required unless PMU event 0x004C is required. If a workaround is needed, this erratum can be avoided by counting three separate PMU events in place of event 0x004C:

- Event 0x0005 (L1D\_TLB\_REFILL)
- Event 0x004D (L1D\_TLB\_REFILL\_WR)
- Event 0x10E. (L1D\_TLB\_REFILL\_RD\_PF)

These events can be used to calculate an Effective event 0x004C as follows:

Effective Event 0x004C = Event 0x0005 - Event 0x004D - Event 0x010E

Effective event 0x004C can be used in place of event 0x004C in calculation of "Attributable Level 1 TLB refill rate, read" to provide an accurate rate calculation.

Arm Architecture Reference Manual relevant events:

Mnemonic	Number
L1D_TLB_REFILL	0x0005
L1D_TLB_REFILL_RD	0x004C
L1D_TLB_REFILL_WR	0x004D
L1D_TLB_RD	0x004E

Implementation Defined relevant event:

Mnemonic	Number
L1D_TLB_REFILL_RD_PF	0x010E

Arm Architecture Reference Manual relevant metric:

"Attributable Level 1 TLB refill rate, read" (Event 0x004C / Event 0x004E)

## 3607348

### PSTATE.{PAN,UAO} synchronization might not be honored while MSR PSTATE is speculative

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0 and r1p1. Open.

#### Description

When software directly writes PSTATE.PAN or PSTATE.UAO with an MSR instruction, the Arm Architecture specifies that side-effects are guaranteed to be visible to later instructions in the Execution stream. However, for a window of time prior to the execution of MSR PSTATE.{PAN,UAO}, instructions following the MSR might speculatively execute with the old context, prior to re-executing non-speculatively under the new, expected context.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

The erratum occurs if the following condition applies:

- MSR PSTATE.{PAN or UAO} executes

#### Implications

Speculative execution of instructions using stale PSTATE.{UAO,PAN} context could in theory present a window of opportunity for a security attack. However, Arm security team has evaluated the practical risk to be very low, given the use-cases of the bits in question and the complexity involved in exploiting.

#### Workaround

A workaround is not expected to be required.

## 3633467

### EDSCR.STATUS not updated on Halting Step when a Load-Exclusive instruction generates a synchronous exception

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0 and r1p1. Open.

#### Description

When a Load-Exclusive instruction is executed with Halting Step enabled, EDSCR.STATUS is not updated if the Load-Exclusive instruction causes a synchronous exception.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

This erratum occurs under the following conditions:

1. In Debug state, the debugger enables Halting Step
2. Debug state is exited and a Load-Exclusive instruction (LDX\*/LDAX\*) is stepped
3. The Load-Exclusive generates a synchronous exception while executing

#### Implications

If the conditions are met, EDSCR.STATUS will not be updated.

#### Workaround

There is no workaround.

## 3700181

### PE might fail to log a RAS error for L2 data RAM ECC errors

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0 and r1p1. Open.

#### Description

Under specific circumstances, the L2 cache might fail to log a corrected or uncorrected ECC error in the PE ERXSTATUS/MISC/ADDR registers.

#### Configurations affected

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

#### Conditions

The erratum occurs if all the following conditions apply:

1. Error correction is enabled with ERROCTL.ED set to 1.
2. PE is performing simultaneous memory reads to both Device or Normal Non-cacheable and Normal-WriteBack memory.
3. Specific timing conditions occur.
4. PE detects an ECC error in the L2 data RAM.

#### Implications

If the specified conditions occur, the PE might not report the ECC error detected by the L2.

Note that there is no silent data corruption - any consumers of the data will receive a poison indication along with the data. The issue is a failure to report the error to the RAS error log.

#### Workaround

No workaround is necessary for this erratum.

## 3705917

### PMU events are mis-categorized by not considering the effect of "Taken locally"

#### Status

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r1p0 and r1p1. Open.

#### Description

FEAT\_VHE establishes broad use of "Taken locally" as a qualifier that determines which instances of an exception are counted by particular PMU events.

PMU events are mis-categorized by failing to consider "Taken locally", specifically resulting in mis-categorizations between PMU events EXC\_UNDEF and EXC\_TRAP\_OTHER, as well as between PMU events EXC\_SVC and EXC\_TRAP\_OTHER.

#### Configurations affected

This erratum affects all configurations.

#### Conditions

The erratum can occur if one of the following conditions apply:

1. When the effective value of HCR\_EL2.{E2H,TGE} **is** {1,1}, an exception can increment PMU event 0x008D EXC\_TRAP\_OTHER, when the exception should instead increment PMU event 0x0081 EXC\_UNDEF.
2. When the effective value of HCR\_EL2.{E2H,TGE} is **NOT** {1,1}, an exception can increment PMU event 0x0081 EXC\_UNDEF, when the exception should instead increment PMU event 0x008D EXC\_TRAP\_OTHER.
3. When the effective value of HCR\_EL2.{E2H,TGE} is **NOT** {1,1}, executing an SVC instruction can increment PMU event 0x0082 EXC\_SVC, when that SVC instruction should instead increment PMU event 0x008D EXC\_TRAP\_OTHER.

#### Implications

When the previous conditions are met, PMU event counts might be inaccurate for events 0x0081, 0x0082, and 0x008D.

#### Workaround

There is no workaround.

# Proprietary notice

This document is protected by copyright and other related rights and the use 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 Limited ("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 the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. 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, any patents, copyrights, trade secrets, trademarks, or other rights.

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.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted 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.

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 this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.



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

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(PRE-1121-V1.0)

# Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in the Arm documents.

## Product status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

### Product completeness status

The information in this document is Final, that is for a developed product.

### Product revision status

The rxpy identifier indicates the revision status of the product described in this manual, where:

**rx**

Identifies the major revision of the product.

**py**

Identifies the minor revision or modification status of the product.