



Arm Cortex-A32 (MP063)

Software Developer Errata Notice

Date of issue: 10-Feb-2023

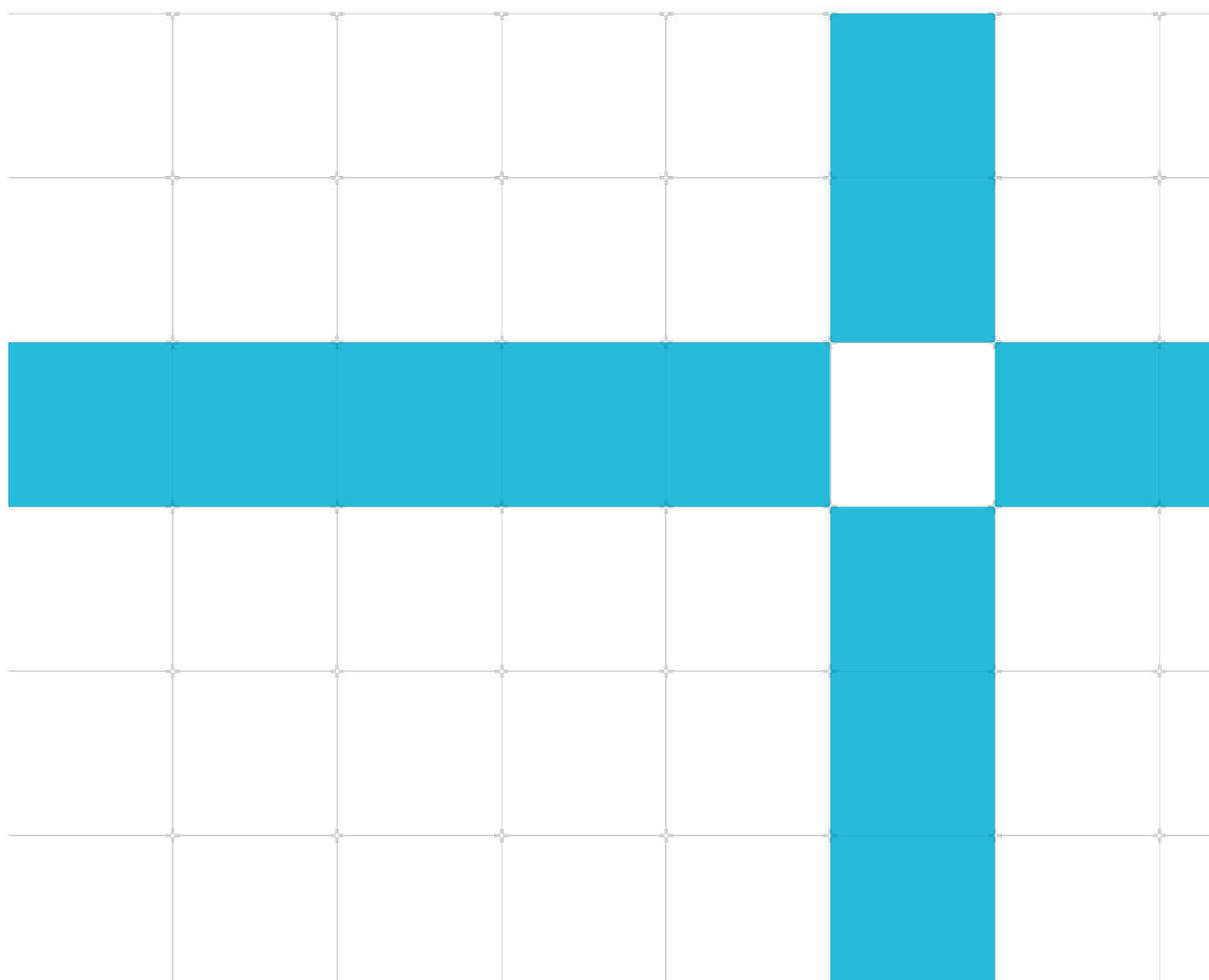
Non-Confidential

Document version: v5.0

Copyright © 2016-2017, 2019, 2023 Arm® Limited (or its affiliates). All rights reserved.

Document ID: SDEN-843852

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



Non-confidential proprietary notice

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

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

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

This document may include technical inaccuracies or typographical errors.

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

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

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

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

Copyright © 2016-2017, 2019, 2023 Arm® Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm Cortex-A32 (MP063), create a ticket on <https://support.developer.arm.com>.

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

Inclusive language commitment

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

If you find offensive language in this document, please email terms@arm.com.

Contents

Introduction	5
Scope	5
Categorization of errata	5
Change Control	6
Errata summary table	7
Errata descriptions	8
Category A	8
Category A (rare)	8
Category B	9
1617552 Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation	9
Category B (rare)	10
Category C	11
2850234 ATB flush response may be delayed	11
1289242 PMU counter might be inaccurate when monitoring BUS_ACCESS and BUS_ACCESS_ST	13
857689 Mismatch between EDPRSR.SR and EDPRSR.R	15
821235 Debug not entering Memory Access mode without setting EDSCR.ERR	16
821232 ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set	17
821227 Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR	18
821221 Instruction cache parity error might cause an incorrect abort to be taken	20
821117 ETM might not generate an event packet and ATB trigger	21
821110 ETM might assert AFREADY before all trace has been output	22
821090 Accessing EDPCSR has side-effects when OS Lock is locked	23
821027 ROM table entries for cores 1, 2, and 3 might be ignored	24

Introduction

Scope

This document describes errata categorized by level of severity. Each description includes:

- The current status of the erratum.
- Where the implementation deviates from the specification and the conditions required for erroneous behavior to occur.
- The implications of the erratum with respect to typical applications.
- The application and limitations of a workaround where possible.

Categorization of errata

Errata are split into three levels of severity and further qualified as common or rare:

Category A	A critical error. No workaround is available or workarounds are impactful. The error is likely to be common for many systems and applications.
Category A (Rare)	A critical error. No workaround is available or workarounds are impactful. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category B	A significant error or a critical error with an acceptable workaround. The error is likely to be common for many systems and applications.
Category B (Rare)	A significant error or a critical error with an acceptable workaround. The error is likely to be rare for most systems and applications. Rare is determined by analysis, verification and usage.
Category C	A minor error.

Change Control

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

10-Feb-2023: Changes in document version v5.0

ID	Status	Area	Category	Summary
2850234	New	Programmer	Category C	ATB flush response may be delayed

23-Oct-2019: Changes in document version v4.0

ID	Status	Area	Category	Summary
1617552	New	Programmer	Category B	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation

28-Feb-2019: Changes in document version v3.0

ID	Status	Area	Category	Summary
1289242	New	Programmer	Category C	PMU counter might be inaccurate when monitoring BUS_ACCESS and BUS_ACCESS_ST

29-Mar-2017: Changes in document version v2.0

ID	Status	Area	Category	Summary
821027	New	Programmer	Category C	ROM table entries for cores 1, 2, and 3 might be ignored
821090	New	Programmer	Category C	Accessing EDPCSR has side-effects when OS Lock is locked
821110	New	Programmer	Category C	ETM might assert AFREADY before all trace has been output
821117	New	Programmer	Category C	ETM might not generate an event packet and ATB trigger
821221	New	Programmer	Category C	Instruction cache parity error might cause an incorrect abort to be taken
821227	New	Programmer	Category C	Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR
821232	New	Programmer	Category C	ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set
821235	New	Programmer	Category C	Debug not entering Memory Access mode without setting EDSCR.ERR
857689	New	Programmer	Category C	Mismatch between EDPRSR.SR and EDPRSR.R

10-Mar-2016: Changes in document version v1.0

No errata in this document version.

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
1617552	Programmer	Category B	Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation	r0p0, r0p1, r1p0	Open
2850234	Programmer	Category C	ATB flush response may be delayed	r0p0, r0p1, r1p0	Open
1289242	Programmer	Category C	PMU counter might be inaccurate when monitoring BUS_ACCESS and BUS_ACCESS_ST	r0p0, r0p1, r1p0	Open
857689	Programmer	Category C	Mismatch between EDPRSR.SR and EDPRSR.R	r0p0, r0p1, r1p0	Open
821235	Programmer	Category C	Debug not entering Memory Access mode without setting EDSCR.ERR	r0p0	r0p1
821232	Programmer	Category C	ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set	r0p0, r0p1, r1p0	Open
821227	Programmer	Category C	Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR	r0p0	r0p1
821221	Programmer	Category C	Instruction cache parity error might cause an incorrect abort to be taken	r0p0	r0p1
821117	Programmer	Category C	ETM might not generate an event packet and ATB trigger	r0p0	r0p1
821110	Programmer	Category C	ETM might assert AFREADY before all trace has been output	r0p0	r0p1
821090	Programmer	Category C	Accessing EDPCSR has side-effects when OS Lock is locked	r0p0	r0p1
821027	Programmer	Category C	ROM table entries for cores 1, 2, and 3 might be ignored	r0p0	r0p1

Errata descriptions

Category A

There are no errata in this category.

Category A (rare)

There are no errata in this category.

Category B

1617552

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

Status

Affects: Cortex-A32

Fault Type: Programmer Category B

Fault Status: Present in r0p0, r0p1, r1p0. Open.

Description

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

Configurations Affected

All configurations are affected.

Conditions

1. A speculative AT instruction performs a table walk, translating a virtual address to a physical address using registers associated with an out-of-context translation regime.
2. Address translation data that is generated during the walk is cached in the 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 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. A workaround is only required if the system software contains an AT instruction as part of an executable page.

Category B (rare)

There are no errata in this category.

Category C

2850234

ATB flush response may be delayed

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r0p1, and r1p0. Open.

Description

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

Configurations Affected

This erratum affects all processor configurations.

Conditions

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

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

Implications

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

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

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

Workaround

There is no workaround to reliably avoid this erratum. As an alternative to waiting for the flush to be acknowledged, a sufficiently long timeout can be used if it is likely that trace generation has stopped. If ATB upsizeers are present in the system, this workaround will not be effective.

1289242

PMU counter might be inaccurate when monitoring BUS_ACCESS and BUS_ACCESS_ST

Status

Affects: Cortex-A32 MPCore

Fault type: Programmer Category C

Fault status: Present in r0p0, r0p1, r1p0. Open.

Description

The Cortex-A32 processor implements a Performance Monitor Unit (PMU). The PMU allows programmers to gather statistics on the operation of the processor during runtime. Because of this erratum, the PMU counter values might be inaccurate when monitoring BUS_ACCESS and BUS_ACCESS_ST events.

Configurations Affected

To be affected by this erratum, one or more of the following must be true:

- The processor is configured with an ACE or CHI bus interface
- The processor is configured with more than one core
- The processor is configured with an L2 cache
- The processor is configured with CPU cache protection

Conditions

1. A performance counter is enabled and configured to count BUS_ACCESS or BUS_ACCESS_ST events.
2. A write or eviction occurs on the bus.

Implications

The PMU counter will erroneously increment or erroneously fail to increment. The inaccuracy varies between cores. Some bus accesses for core 0 might be attributed to core 1. Some bus accesses for core 1 might be attributed to core 2 or core 3. Bus accesses for cores 2 and 3 will not be attributed to any core. The number of cores present in the configuration does not affect how the bus accesses are attributed. For example, some core 1 accesses might be attributed to cores 2 and 3 even if the configuration has only two cores. The impact on the final counter value in each core is high.

This might lead to inaccurate results when using the PMU to debug or profile code.

Workaround

The BUS_ACCESS and BUS_ACCESS_ST events can be counted accurately for core 0 by enabling the counter on both core 0 and core 1 and taking the total. This workaround requires core 1 to be present in the configuration and idle during testing. Similarly, the total count for core 0 and core 1 can be found by enabling the counter on all four cores and taking the total. This workaround requires cores 2 and 3 to be present in the configuration and idle during testing. In a configuration with four cores, the events can be used with an intensive memory test that runs on two cores to give a reasonable indication of maximum bandwidth for the cluster.

The event L2D_CACHE_WB counts write-backs from the L2 cache that are attributable to a core. For a test focused on cacheable memory bandwidth, this might be a suitable replacement for BUS_ACCESS_ST. This event includes:

- Write-backs as a result of evictions and cache maintenance instructions.
- Writes in read allocate mode (write-streaming mode).

L2D_CACHE_WB does not include:

- Write-backs as a result of snoop requests from outside of the cluster.
- Write-backs as a result of ACP interface accesses.

857689

Mismatch between EDPRSR.SR and EDPRSR.R

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r0p1, r1p0. Open.

Description

The processor provides access to the EDPRSR through the APB interface. If this access is done at the same time as the core leaves a Warm reset, then a subsequent read of the same register will read an incorrect value of the SR field.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

1. The core is in Warm reset.
2. A debugger reads the EDPRSR register over the APB interface.
3. The core comes out of Warm reset during the APB read.
4. A second APB read is made to the EDPRSR register.

Implications

The first read of the EDPRSR will read the SR field and R field both as 0b1.

The second read of the EDPRSR.SR field will read 0b0 whereas the previous read of the EDPRSR.SR was 0b1 while in Warm reset. Because the first read took place while in Warm reset, the sticky bit should still be set on the second read.

Workaround

If the debugger reads the EDPRSR and sees both the SR and R fields set, then it must remember this result and on the next EDPRSR read treat the SR bit as if it was set.

821235

Debug not entering Memory Access mode without setting EDSCR.ERR

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

The processor supports entering debug Memory Access (MA) mode through the APB interface. This might fail when an instruction previously executed through the APB interface has not yet completed.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

1. The debugger inserts an instruction through the APB interface by writing the EDITR.
2. The debugger writes to the EDSCR to enter MA mode.
3. The debugger reads the DTRTX at the same time as the instruction completes.

Implications

The APB read of the DTRTX does not start up the MA state machine, but EDSCR.TXU and EDSCR.ERR are not set.

Workaround

Before executing the instruction, the debugger should write 1 to the EDRCR.CSPA. After executing the instruction, the debugger should poll the EDSCR.PipeAdv bit to determine when the instruction has completed, and only attempt to enter MA mode after the PipeAdv bit is set.

821232

ATS12NSOPR instruction might incorrectly translate when the HCR.TGE bit is set

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r0p1, r1p0. Open.

Description

An ATS12NSOPR address translation instruction executed from EL3 might report an incorrect result in the PAR when both the HCR.TGE bit is set and the SCTLR.M is set.

Configurations Affected

This erratum affects all configurations of the processor.

Conditions

1. The core is executing at Exception level 3 in AArch32.
2. SCR.NS = 0
3. HCR.TGE = 1
4. SCTLR(ns).M = 1
5. The core executes an ATS12NSOPR instruction

Implications

The PAR register will incorrectly report the translation as if the stage 1 MMU was enabled. Note that the combination of both HCR.TGE and SCTLR.M bits being set was UNPREDICTABLE in earlier versions of the architecture, and was only given a defined behavior to reduce the UNPREDICTABLE space. It is not expected to be a useful combination for software.

Workaround

Secure software can clear the SCTLR(ns).M bit before executing the address translation instruction, if the HCR.TGE bit is set. It should restore the previous SCTLR(ns).M value before returning to a lower Exception level.

821227

Incorrect fault status codes used for reporting synchronous uncorrectable ECC aborts in the IFSR

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

When reporting a synchronous uncorrectable ECC error, an incorrect encoding is written to the fault status code field of the Instruction Fault Status Register.

Configurations Affected

This erratum affects configurations of the processor that have CPU_CACHE_PROTECTION or SCU_CACHE_PROTECTION set to TRUE.

Conditions

1. A core is in the AArch32 execution state.
2. The core is executing in EL0, EL1, or EL3.
3. The core is using the Short-descriptor page format.
4. An uncorrectable ECC error occurs in one of the following caches:
 - The L1 data cache of the core.
 - The L1 data cache of another core in the processor.
 - The L2 cache.
5. An instruction fetch or a translation table walk for an instruction fetch hits the cache line containing the ECC error, triggering a prefetch abort.

If these conditions are met, then IFSR.FS[4] is set to 0 instead of 1.

Implications

When the short descriptor page format is in use, software cannot make a distinction between a Domain fault, level 1, or a synchronous parity or ECC error on memory access. Both will be reported in the IFSR as 01001.

Similarly, software cannot make a distinction between synchronous external abort on translation table walk and synchronous parity or ECC error on translation table walk. Both will be reported in the IFSR as synchronous external abort on translation table walk (IFS_R.FS[4:0] will be 01100 for translation level 1 and 01110 for translation level 2). This is valid behavior because it is IMPLEMENTATION DEFINED whether ECC errors are reported using the assigned fault code or using another appropriate encoding. However, the behavior is inconsistent with fault code usage in other fault status registers, and in the IFSR when using the Long-descriptor page format.

Workaround

When reading the fault status code from the IFSR, software will have to be aware that the value reported might not be accurate. If software reads 01001 for IFS_R.FS[4:0], then software will have to determine if a Domain fault could have occurred for this address.

821221

Instruction cache parity error might cause an incorrect abort to be taken

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

The core cache protection option allows the processor to detect and correct a 1-bit error in the instruction cache RAMs. If an error is detected on an instruction at the end of a page, and the following page would generate a fault if executed, then under some conditions it is possible for a fault to be taken when no instruction from the faulting page was architecturally executed.

Configurations affected

This erratum affects configurations of the processor with CPU_CACHE_PROTECTION enabled.

Conditions

1. The core is executing in AArch32 T32 instruction state.
2. The program counter points to a 16-bit length instruction which is at the end of a page.
3. The related instruction is stored in the L1 instruction cache.
4. There is a single bit error on one specific bit of the instruction, which causes the instruction to be incorrectly interpreted as a 32-bit instruction.
5. The page which is immediately after the address range of the previous 16-bit instruction would cause a translation fault, access flag fault, or permission fault if it was accessed.

Implications

If the above conditions are met, then the translation, permission, or access flag fault is incorrectly taken, instead of correcting the error. There is still substantial benefit being gained from the parity logic. There might be a negligible increase in overall system failure rate due to this erratum.

Workaround

No workaround is required.

821117

ETM might not generate an event packet and ATB trigger

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0. fixed in r0p1.

Description

The ETM contains four resources to generate events based on core activity. When the ETM generates an event, it is reported to the CTI on the external outputs bus, an ATB trigger is generated, and an event packet is inserted into the trace stream.

Because of this errata, when the ETM is becoming idle, there is a one cycle window where an event might get indicated to the CTI, but would not generate an ATB trigger and would not generate an event packet.

Configurations Affected

To be affected by this erratum, the processor must be configured with ETMs.

Conditions

1. The ETM is becoming idle due to any of the following
 - The ETM has been disabled due to TRCPRGCTLR.EN = 0 or TRCOSLAR.OSLK = 1
 - The core has started to execute a WFI or WFE and is going to enter sleep
 - **DBGEN/NIDEN** have changed and trace is no longer permitted
2. The resources are configured to generate events.
3. An event is generated because of a PMU event or CTI trigger on the last cycle in which the ETM could generate an event.

Implications

If the stimulus that caused the event to be generated occurred one cycle later, then the event would not have been generated at all. Therefore, this errata will only be noticed when trying to correlate the CTI behaviour with the trace stream.

Workaround

There is no workaround for this erratum.

821110

ETM might assert **AFREADY** before all trace has been output

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

When the **AFVALID** signal on the ATB interface is asserted, the ETM should immediately start outputting all buffered trace. It should assert the **AFREADY** output one cycle after all trace that was buffered on the cycle in which **AFVALID** was first asserted has been output.

Because of this erratum, the **AFREADY** signal might be asserted before all the necessary trace has been output.

Configurations Affected

To be affected by this erratum, the processor must be configured with ETMs.

Conditions

1. The ETM must contain buffered trace.
2. **ATVALID** must be LOW on the first cycle **AFVALID** is HIGH.

Implications

This might result in the ETM containing trace that was generated before the flush request when the rest of the system expects this trace to have been output.

Workaround

The system can ensure that all trace has been drained from the ETM by disabling it. The ETM can be disabled by setting TRCPRGCTLR.EN to 0. The system should then poll the TRCSTATR.IDLE bit. When it reads as 1, the ETM is idle and all trace that was generated before the write to TRCPRGCTLR has been output.

821090

Accessing EDPCSR has side-effects when OS Lock is locked

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

Access to the External Debug Program Counter Sample Register, EDPCSR, should not have side-effects when the OS lock is locked or when the OS Double Lock is locked. Because of this erratum, the side-effects are only disabled when OS Double Lock is locked.

Configurations Affected

All configurations are affected.

Conditions

1. The OS Lock is locked (EDPRSR.OSLK == 1).
2. The OS Double Lock is unlocked (EDPRSR.DLK == 0).
3. An external debugger reads EDPCSR[31:0].

Implications

If the erratum conditions are met then EDCIDSR and EDVIDSR will be updated as if the OS Lock were unlocked.

Workaround

There is no workaround.

821027

ROM table entries for cores 1, 2, and 3 might be ignored

Status

Affects: Cortex-A32

Fault Type: Programmer Category C

Fault Status: Present in r0p0. Fixed in r0p1.

Description

The processor includes a ROM table that allows external debuggers to determine which debug components are implemented inside the processor. The end of the ROM table is specified by an end-marker that reads 0x00000000.

Because of this erratum, the ROM table might include an end-marker before the end of the ROM table.

Configurations Affected

To be affected by this erratum, all the following must be true:

- The processor is configured with more than one core.
- The processor is configured with the v8 debug map (not the legacy v7 debug map).
- The processor is configured without ETMs.

Conditions

1. An external debugger reads the ROM entry register for the core 0 ETM component.
2. The external debugger interprets the read value of 0x00000000 as the ROM table end-marker for all cores in the processor.

Implications

The external debugger might report core 1, core 2, and core 3 debug components as not present even if the cores are present. This might prevent the user from accessing those components with the debugger.

Workaround

The external debugger must read the ROM entry registers for a core even if it reads an end-marker in the ROM entry register for the previous core's ETM component.

