# Arm Cortex-A34 (MP066)

# Software Developer Errata Notice

**Date of issue:** 10-Feb-2023

**Non-Confidential**

**Document version:** v5.0

**Document ID:** SDEN-843855

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

# Contents

# Introduction

## Scope

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

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

## Categorization of errata

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

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

# Change Control

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

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

| ID | Status | Area | Category | Summary |
|---|---|---|---|---|
| **2850235** | New | Programmer | Category C | ATB flush response may be delayed |

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

| ID | Status | Area | Category | Summary |
|---|---|---|---|---|
| **1617549** | 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 |
|---|---|---|---|---|
| **1289240** | 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 |
|---|---|---|---|---|
| **821105** | New | Programmer | Category B | Clearing TCR_ELx.TBI from inside a tagged address region might cause a Translation fault or an Address size fault |
| **821026** | New | Programmer | Category C | ROM table entries for cores 1, 2, and 3 might be ignored |
| **821073** | New | Programmer | Category C | ETM might output an incorrect exception return address |
| **821087** | New | Programmer | Category C | WFx can cause the PC to jump from lower VA subrange to upper VA subrange |
| **821089** | New | Programmer | Category C | Accessing EDPCSR has side-effects when OS Lock is locked |
| **821108** | New | Programmer | Category C | ETM might assert AFREADY before all trace has been output |
| **821116** | New | Programmer | Category C | ETM might not generate an event packet and ATB trigger |
| **821230** | New | Programmer | Category C | ETM trace reports incorrect branch target |
| **821233** | New | Programmer | Category C | Debug not entering Memory Access mode without setting EDSCR.ERR |
| **857687** | 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 |
|---|---|---|---|---|---|
| **1617549** | Programmer | Category B | Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation | r0p0, r0p1 | Open |
| **821105** | Programmer | Category B | Clearing TCR_ELx.TBI from inside a tagged address region might cause a Translation fault or an Address size fault | r0p0 | r0p1 |
| **2850235** | Programmer | Category C | ATB flush response may be delayed | r0p0, r0p1 | Open |
| **1289240** | Programmer | Category C | PMU counter might be inaccurate when monitoring BUS_ACCESS and BUS_ACCESS_ST | r0p0, r0p1 | Open |
| **857687** | Programmer | Category C | Mismatch between EDPRSR.SR and EDPRSR.R | r0p0, r0p1 | Open |
| **821233** | Programmer | Category C | Debug not entering Memory Access mode without setting EDSCR.ERR | r0p0 | r0p1 |
| **821230** | Programmer | Category C | ETM trace reports incorrect branch target | r0p0 | r0p1 |
| **821116** | Programmer | Category C | ETM might not generate an event packet and ATB trigger | r0p0 | r0p1 |
| **821108** | Programmer | Category C | ETM might assert AFREADY before all trace has been output | r0p0 | r0p1 |
| **821089** | Programmer | Category C | Accessing EDPCSR has side-effects when OS Lock is locked | r0p0 | r0p1 |
| **821087** | Programmer | Category C | WFx can cause the PC to jump from lower VA subrange to upper VA subrange | r0p0 | r0p1 |
| **821073** | Programmer | Category C | ETM might output an incorrect exception return address | r0p0 | r0p1 |
| **821026** | 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

## 1617549
## Speculative AT instruction using out-of-context translation regime could cause subsequent request to generate an incorrect translation

### Status

Affects: Cortex-A34
Fault Type: Programmer Category B
Fault Status: Present in r0p0, r0p1. 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.

## 821105
## Clearing TCR_ELx.TBI from inside a tagged address region might cause a Translation fault or an Address size fault

### Status

Affects: Cortex-A34
Fault Type: Programmer Category B
Fault Status: Present in r0p0. Fixed in r0p1.

### Description

The ARMv8 architecture supports address tagging in the AArch64 execution state. When address tagging is used, the top byte of the virtual address is ignored during address translation. If a branch is executed, tag bits should be removed from the target address when the program counter is updated. Because of this erratum, the tag bits might propagate to the program counter. If the Top Byte Ignored bit is subsequently cleared in the relevant Translation Control Register, then the processor might report a fault because the program counter is out of range.

### Configurations Affected

All configurations are affected.

### Conditions

1. The TBI bit in the Translation Control Register for the current Exception level is set to 1.
2. The core executes an indirect branch to an address where the top byte is not sign-extended.
3. The branch was incorrectly predicted not-taken before its execution.

### Implications

If the above conditions are met, then the address tag might propagate to an internal copy of the program counter. Reads of the PC register are not affected.

If the TBI bit remains set to 1, then the address tag will be removed from the program counter at the next context synchronization event. Address translations for instruction fetches will continue to ignore the top byte of the virtual address and no unexpected behavior will occur.

If the TBI bit is cleared to 0 by the code in the address-tagged region, and if the change takes effect before or as part of the next context synchronization event, then the core might report a fault because the program counter is out of range.

For example, if an MSR instruction to clear the TBI bit is followed by an ISB instruction, then successful execution of the ISB will resolve the issue. However, if the change in TBI bit takes effect before the ISB is executed then the address translation that is performed to fetch the ISB instruction might trigger a Translation fault (if the translation system is enabled) or an Address size fault (if the translation system is not enabled).

### Workaround

The issue can be avoided by inserting an ISB instruction before the instruction that clears the TBI bit.

## Category B (rare)

There are no errata in this category.

# Category C

## 2850235
## ATB flush response may be delayed

### Status

Affects: Cortex-A34
Fault Type: Programmer Category C
Fault Status: Present in r0p0, and r0p1. Open.

### Description

The *Embedded Trace Macrocell* (ETM) supports an external flush request for each ATB bus. 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 upsizers are present in the system, this workaround will not be effective.

## 1289240
## PMU counter might be inaccurate when monitoring BUS_ACCESS and BUS_ACCESS_ST

### Status

Affects: Cortex-A34 MPCore
Fault type: Programmer Category C
Fault status: Present in r0p0, r0p1. Open.

### Description

The Cortex-A34 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.

## 857687
## Mismatch between EDPRSR.SR and EDPRSR.R

### Status

Affects: Cortex-A34
Fault Type: Programmer Category C
Fault Status: Present in r0p0, r0p1. 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.

## 821233
## Debug not entering Memory Access mode without setting EDSCR.ERR

### Status

Affects: Cortex-A34
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.

## 821230
## ETM trace reports incorrect branch target

### Status

Affects: Cortex-A34
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

### Description

When executing a branch that targets an out of range virtual address, the ETM might trace the target address incorrectly.

### Configurations Affected

This erratum affects all configurations of the processor.

### Conditions

1. ETM trace is enabled and not prohibited.
2. The core is executing code located in the virtual address range 0hFFFF000000000000-0hFFFFFFFFFFFFFFFF, and bit[48] of the PC is set. If the relevant TCR.TBI bit is set, then bits[63:56] of the virtual address can be any value.
3. The core executes a branch instruction with the target virtual address in the range 0h0002000000000000-0hFFFEFFFFFFFFFFFF, with bit[48] of the target address clear. This could be any kind of instruction that alters the program flow, including immediate branches, register based branches, and returns.

### Implications

The target of the branch will take a translation fault, because it is not in a valid address range, but because the ETM trace decompression will report the branch target incorrectly the reason for the fault might be harder to determine when debugging. Bits[47:0] of the branch target will still be correct, but bits[63:48] will be incorrectly reported as all zero.

### Workaround

There is no workaround for this erratum.

## 821116
## ETM might not generate an event packet and ATB trigger

### Status

Affects: Cortex-A34
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.

## 821108
## ETM might assert AFREADY before all trace has been output

### Status

Affects: Cortex-A34
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.

## 821089
## Accessing EDPCSR has side-effects when OS Lock is locked

### Status

Affects: Cortex-A34
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, EDVIDSR, and EDPCSR[63:32] will be updated as if the OS Lock were unlocked.

### Workaround

There is no workaround.

## 821087
## WFx can cause the PC to jump from lower VA subrange to upper VA subrange

Affects: Cortex-A34
Fault Type: Programmer Category C
Fault Status: Present in r0p0. Fixed in r0p1.

### Description

If a core executes a WFI or WFE instruction at VA 0x0000_ffff_ffff_fffc in the AArch64 execution state then, when the core wakes up, it should resume execution from VA 0x0001_0000_0000_0000. This address is outside of the implemented virtual address space and will trigger a Translation fault if accessed.

Because of this erratum, the core will resume execution from VA 0xffff_0000_0000_0000 instead of 0x0001_0000_0000_0000. Address 0xffff_0000_0000_0000 is a valid address in the upper VA subrange for EL0/EL1 translations if TCR_EL1.T1SZ is set to 16.

### Configurations Affected

All configurations.

### Conditions

1. The core executes a WFI or WFE instruction at VA 0x0000_ffff_ffff_fffc.

### Implications

Real code is not expected to include a WFI or WFE instruction at VA 0x0000_ffff_ffff_fffc because, if this erratum did not exist, wake-up from the WFx would always trigger an access to a VA that is outside of the implemented virtual address space.

### Workaround

If a workaround is required, code could be inserted at VA 0xffff_0000_0000_0000 to catch unexpected accesses to this address.

## 821073
## ETM might output an incorrect exception return address

### Status

Affects: Cortex-A34
Fault Type: Programmer Cat-C
Fault Status: Present in r0p0. Fixed in r0p1.

### Description

The processor supports instruction trace using a per-core ETM. If tracing is enabled, a core's ETM generates an Exception element when the core takes an exception. The Exception element contains an indication that an exception has occurred, the type of exception, and the exception return address. Because of this erratum, the exception return address that is output by the ETM might have bits [63:49] set to 1 when they should be set to 0.

### Configurations Affected

This erratum affects configurations of the processor with a per-core ETM.

### Conditions

1. Trace is enabled.
2. Non-invasive debug is permitted.
3. The core executes one of the following instructions at VA 0x0000_ffff_ffff_fffc
    - SVC
    - HVC
    - SMC

### Implications

The exception return address reported by the ETM might be incorrect. For example, an SMC instruction at VA 0x0000_ffff_ffff_fffc might cause the ETM to output an exception return address of 0xffff_0000_0000_0000 instead of 0x0001_0000_0000_0000.

Real code is not expected to meet the conditions in this erratum because, if this erratum did not exist, the core would access a VA that is outside of the implemented virtual address space.

### Workaround

There is no workaround for this erratum.

## 821026
## ROM table entries for cores 1, 2, and 3 might be ignored

### Status

Affects: Cortex-A34
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.