



Cortex-M7 (AT610) and Cortex-M7 with FPU (AT611)

Software Developer Errata Notice

Date of issue: May 28, 2024

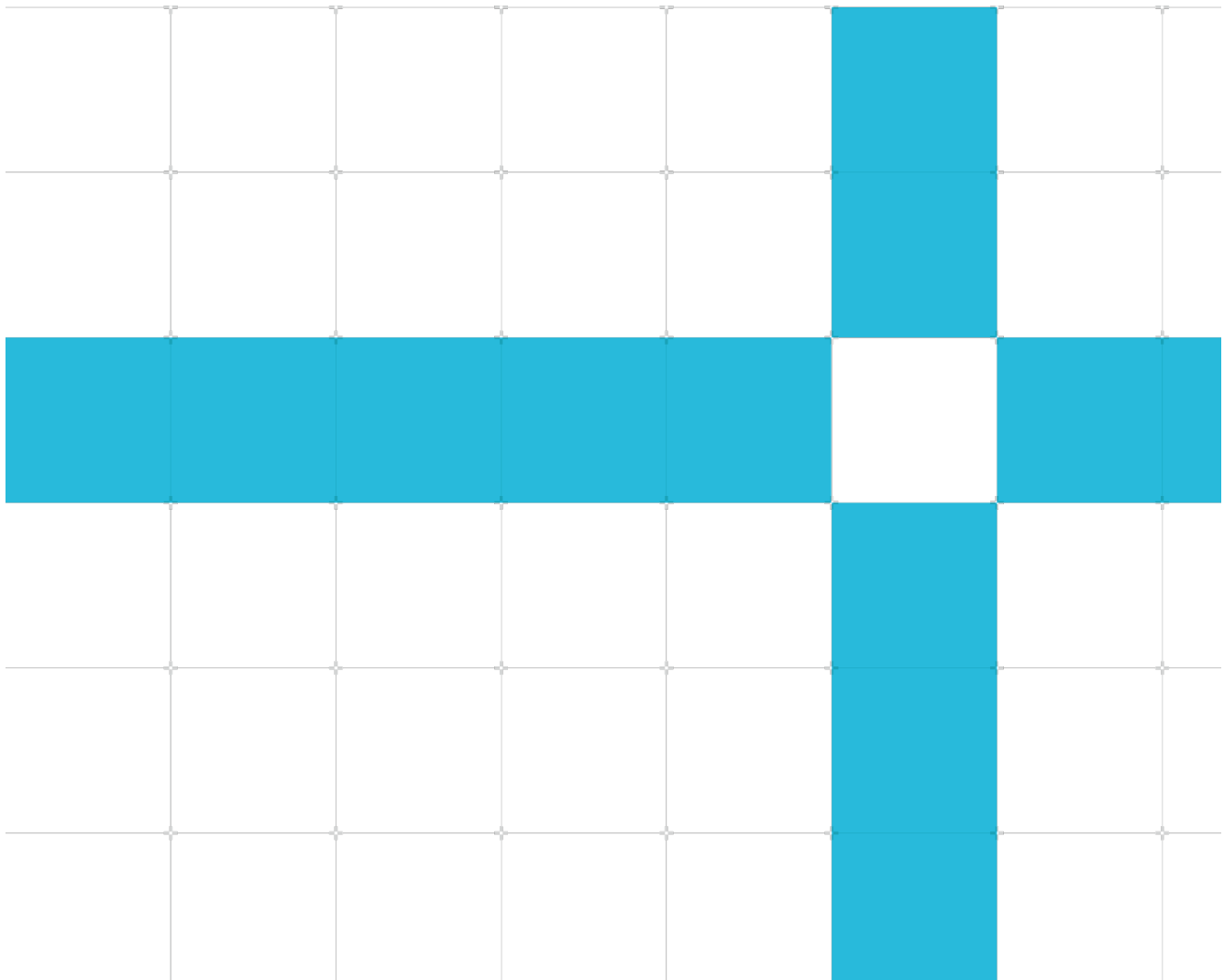
Non-Confidential

Document version: 11.0

Copyright © 2014, 2015, 2018, 2019, 2021, 2024 Arm® Limited (or its affiliates). All rights reserved.

Document ID: SDEN-1068427

This document contains all known errata since the r0p1 release of the product.



This document is Non-Confidential.

Copyright © 2014, 2015, 2018, 2019, 2021, 2024 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_1068427_11.0_en) was issued on May 28, 2024.

There might be a later issue at <http://developer.arm.com/documentation/SDEN-1068427>

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Cortex-M7 (AT610) and Cortex-M7 with FPU (AT611), 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

Introduction	5
Scope	5
Categorization of errata	5
Change Control	6
Errata summary table	9
Errata descriptions	11
Category A	11
1259864 Data corruption in a sequence of Write-Through stores and loads	11
Category A (rare)	12
Category B	13
412512 Cortex-M7 TPIU might lose trace data in configurations with an ETM but no ITM	13
426115 Interrupting a FAULTMASK-setting instruction might cause incorrect MPU instruction attributes	15
440977 Increasing priority using a write to BASEPRI does not take effect immediately	17
565285 Core can send AXI transactions that permit reordering when it should not	19
1013783 PLD might perform linefill to address that would generate a MemManage Fault	21
2328489 TCM bandwidth sharing between AHBS writes and software stores might not function correctly when using TCM wait states	23
Category B (rare)	25
443753 A sequence of cacheable stores to memory locations that always return bus faults might cause deadlock	25
Category C	27
399743 The Fault Address Register (FAR) might be corrupted when BFHFNMIEN is set	27
408519 Incorrect GTS packet generation when global timestamps are enabled during debug using the ITM	29
416915 HFSR.FORCED bit is not set for configurable priority faults which result in LOCKUP	31
421025 Early forwarding from load is incorrectly cancelled inside IT block	33
422825 MPU fetch attributes might transiently be incorrect after an exception return	35
423541 Interrupts on a bus-aborting strongly-ordered or device load to the stack pointer might cause incorrect exception stacking	37
431216 Unimplemented bits of BASEPRI do not read-as-zero	39
449383 Write to FPCCR.ASPEN while a Single-precision FP MAC is completing might corrupt the FP register bank	41
486321 Incorrect behavior of profiling counters	43
505438 TPIU cannot be flushed in Debug state if Cortex-M7 TPIU is used	45

513195	Lock Status Indication incorrectly reads as one for debugger reads	47
636315	Software programming errors might not be reported for on-line MBIST access to the I-Cache	49
702596	Single stepping Cortex-M7 enters pending exception handler	51
1267980	ECC error causes data corruption when the data cache error bank registers are locked	53
1313001	Store after cache invalidate without intervening barrier might cause inconsistent memory view	54
1315869	Data corruption for load following Store-Exclusive	56
1518990	Value used for DWT Data Value Comparison is in memory-endianness format, not little-endian	58
3092511	Cortex-M7 can halt in an incorrect address when breakpoint and exception occurs simultaneously	59
Proprietary notice		61
Product and document information		63
Product status		63
Product completeness status		63
Product revision status		63

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.

May 28, 2024: Changes in document version v11.0

ID	Status	Area	Category	Summary
412512	Updated	Programmer	Category B	Cortex-M7 TPIU might lose trace data in configurations with an ETM but no ITM
426115	Updated	Programmer	Category B	Interrupting a FAULTMASK-setting instruction might cause incorrect MPU instruction attributes
440977	Updated	Programmer	Category B	Increasing priority using a write to BASEPRI does not take effect immediately
565285	Updated	Programmer	Category B	Core can send AXI transactions that permit reordering when it should not
443753	Updated	Programmer	Category B (rare)	A sequence of cacheable stores to memory locations that always return bus faults might cause deadlock
399743	Updated	Programmer	Category C	The Fault Address Register (FAR) might be corrupted when BFHFNMIGN is set
408519	Updated	Programmer	Category C	Incorrect GTS packet generation when global timestamps are enabled during debug using the ITM
416915	Updated	Programmer	Category C	HFSR.FORCED bit is not set for configurable priority faults which result in LOCKUP
421025	Updated	Programmer	Category C	Early forwarding from load is incorrectly cancelled inside IT block
422825	Updated	Programmer	Category C	MPU fetch attributes might transiently be incorrect after an exception return
423541	Updated	Programmer	Category C	Interrupts on a bus-aborting strongly-ordered or device load to the stack pointer might cause incorrect exception stacking
431216	Updated	Programmer	Category C	Unimplemented bits of BASEPRI do not read-as-zero
449383	Updated	Programmer	Category C	Write to FPCCR.ASPEN while a Single-precision FP MAC is completing might corrupt the FP register bank
486321	Updated	Programmer	Category C	Incorrect behavior of profiling counters
505438	Updated	Programmer	Category C	TPIU cannot be flushed in Debug state if Cortex-M7 TPIU is used
513195	Updated	Programmer	Category C	Lock Status Indication incorrectly reads as one for debugger reads
3092511	New	Programmer	Category C	Cortex-M7 can halt in an incorrect address when breakpoint and exception occurs simultaneously

October 25, 2021: Changes in document version v10.0

ID	Status	Area	Category	Summary
2328489	New	Programmer	Category B	TCM bandwidth sharing between AHBS writes and software stores might not function correctly when using TCM wait states

December 04, 2019: Changes in document version v9.0

ID	Status	Area	Category	Summary
1518990	New	Programmer	Category C	Value used for DWT Data Value Comparison is in memory-endianness format, not little-endian

November 28, 2018: Changes in document version v8.0

ID	Status	Area	Category	Summary
1013783	Updated	Programmer	Category B	PLD might perform linefill to address that would generate a MemManage Fault
1267980	New	Programmer	Category C	ECC error causes data corruption when the data cache error bank registers are locked
1313001	New	Programmer	Category C	Store after cache invalidate without intervening barrier might cause inconsistent memory view
1315869	New	Programmer	Category C	Data corruption for load following Store-Exclusive

November 09, 2018: Changes in document version v7.0

ID	Status	Area	Category	Summary
1259864	New	Programmer	Category A	Data corruption in a sequence of Write-Through stores and loads
565285	New	Programmer	Category B	Core can send AXI transactions that permit reordering when it should not
1013783	New	Programmer	Category B	PLD might perform linefill to address that would generate a MemManage Fault
636315	New	Programmer	Category C	Software programming errors might not be reported for on-line MBIST access to the I-Cache
702596	New	Programmer	Category C	Single stepping Cortex-M7 enters pending exception handler

July 16, 2015: Changes in document version v6.0

ID	Status	Area	Category	Summary
443753	Updated	Programmer	Category B (rare)	A sequence of cacheable stores to memory locations that always return bus faults might cause deadlock
416915	Updated	Programmer	Category C	HFSR.FORCED bit is not set for configurable priority faults which result in LOCKUP
421025	Updated	Programmer	Category C	Early forwarding from load is incorrectly cancelled inside IT block
486321	New	Programmer	Category C	Incorrect behavior of profiling counters
505438	New	Programmer	Category C	TPIU cannot be flushed in Debug state if Cortex-M7 TPIU is used
513195	New	Programmer	Category C	Lock Status Indication incorrectly reads as one for debugger reads

March 29, 2015: Changes in document version v5.0

No new or updated errata in this document version.

December 04, 2014: Changes in document version v4.0

ID	Status	Area	Category	Summary
412512	New	Programmer	Category B	Cortex-M7 TPIU might lose trace data in configurations with an ETM but no ITM
440977	New	Programmer	Category B	Increasing priority using a write to BASEPRI does not take effect immediately
443753	New	Programmer	Category B (rare)	A sequence of cacheable stores to memory locations that always return bus faults might cause deadlock
408519	New	Programmer	Category C	Incorrect GTS packet generation when global timestamps are enabled during debug using the ITM
449383	New	Programmer	Category C	Write to FPCCR.ASPEN while a Single-precision FP MAC is completing might corrupt the FP register bank

November 11, 2014: Changes in document version v3.0

ID	Status	Area	Category	Summary
426115	New	Programmer	Category B	Interrupting a FAULTMASK-setting instruction might cause incorrect MPU instruction attributes
416915	New	Programmer	Category C	HFSR.FORCED bit is not set for configurable priority faults which result in LOCKUP
422825	New	Programmer	Category C	MPU fetch attributes might transiently be incorrect after an exception return
423541	New	Programmer	Category C	Interrupts on a bus-aborting strongly-ordered or device load to the stack pointer might cause incorrect exception stacking
431216	New	Programmer	Category C	Unimplemented bits of BASEPRI do not read-as-zero

September 12, 2014: Changes in document version v2.0

ID	Status	Area	Category	Summary
399743	New	Programmer	Category C	The Fault Address Register (FAR) might be corrupted when BFHFNMIEN is set
421025	New	Programmer	Category C	Early forwarding from load is incorrectly cancelled inside IT block

April 28, 2014: 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
1259864	Programmer	Category A	Data corruption in a sequence of Write-Through stores and loads	r0p1, r0p2, r1p0, r1p1	r1p2
412512	Programmer	Category B	Cortex-M7 TPIU might lose trace data in configurations with an ETM but no ITM	r0p1	r0p2
426115	Programmer	Category B	Interrupting a FAULTMASK-setting instruction might cause incorrect MPU instruction attributes	r0p1	r0p2
440977	Programmer	Category B	Increasing priority using a write to BASEPRI does not take effect immediately	r0p1	r0p2
565285	Programmer	Category B	Core can send AXI transactions that permit reordering when it should not	r0p1, r0p2, r1p0, r1p1, r1p2	Open
1013783	Programmer	Category B	PLD might perform linefill to address that would generate a MemManage Fault	r0p1, r0p2, r1p0, r1p1, r1p2	Open
2328489	Programmer	Category B	TCM bandwidth sharing between AHBS writes and software stores might not function correctly when using TCM wait states	r0p1, r0p2, r1p0, r1p1, r1p2	Open
443753	Programmer	Category B (rare)	A sequence of cacheable stores to memory locations that always return bus faults might cause deadlock	r0p1	r0p2
399743	Programmer	Category C	The Fault Address Register (FAR) might be corrupted when BFHFNMIEN is set	r0p1	r0p2
408519	Programmer	Category C	Incorrect GTS packet generation when global timestamps are enabled during debug using the ITM	r0p1	r0p2
416915	Programmer	Category C	HFSR.FORCED bit is not set for configurable priority faults which result in LOCKUP	r0p1	r0p2
421025	Programmer	Category C	Early forwarding from load is incorrectly cancelled inside IT block	r0p1	r0p2

ID	Area	Category	Summary	Found in versions	Fixed in version
422825	Programmer	Category C	MPU fetch attributes might transiently be incorrect after an exception return	r0p1	r0p2
423541	Programmer	Category C	Interrupts on a bus-aborting strongly-ordered or device load to the stack pointer might cause incorrect exception stacking	r0p1	r0p2
431216	Programmer	Category C	Unimplemented bits of BASEPRI do not read-as-zero	r0p1	r0p2
449383	Programmer	Category C	Write to FPCCR.ASPEN while a Single-precision FP MAC is completing might corrupt the FP register bank	r0p1	r0p2
486321	Programmer	Category C	Incorrect behavior of profiling counters	r0p1, r0p2, r1p0	r1p1
505438	Programmer	Category C	TPIU cannot be flushed in Debug state if Cortex-M7 TPIU is used	r0p2, r1p0	r1p1
513195	Programmer	Category C	Lock Status Indication incorrectly reads as one for debugger reads	r0p1, r0p2, r1p0	r1p1
636315	Programmer	Category C	Software programming errors might not be reported for on-line MBIST access to the I-Cache	r0p1, r0p2, r1p0, r1p1, r1p2	Open
702596	Programmer	Category C	Single stepping Cortex-M7 enters pending exception handler	r0p1	r0p2
1267980	Programmer	Category C	ECC error causes data corruption when the data cache error bank registers are locked	r0p1, r0p2, r1p0, r1p1, r1p2	Open
1313001	Programmer	Category C	Store after cache invalidate without intervening barrier might cause inconsistent memory view	r0p1, r0p2, r1p0, r1p1, r1p2	Open
1315869	Programmer	Category C	Data corruption for load following Store-Exclusive	r0p1, r0p2, r1p0, r1p1, r1p2	Open
1518990	Programmer	Category C	Value used for DWT Data Value Comparison is in memory-endianness format, not little-endian	r0p1, r0p2, r1p0, r1p1, r1p2	Open
3092511	Programmer	Category C	Cortex-M7 can halt in an incorrect address when breakpoint and exception occurs simultaneously	REL, r0px, r1p0, r1p1	Open

Errata descriptions

Category A

1259864

Data corruption in a sequence of Write-Through stores and loads

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category A

Fault Status: Present in r0p1, r0p2, r1p0 and r1p1. Fixed in r1p2.

Description

If a particular sequence of stores and loads is performed to Write-Through memory, and some timing-based internal conditions are met, then a load might not get the last data stored to that address.

Configurations Affected

All configurations with a data cache are affected.

Conditions

This erratum can only occur if the loads and stores are to Write-Through memory. This could be because of any of the following:

1. The *Memory Protection Unit* (MPU) has been programmed to set this address as Write-Through.
2. The default memory map is being used, and this address is Write-Through in the default memory map.
3. The memory is cacheable, and the CM7_CACR.FORCEWT bit is set.
4. The memory is cacheable, shared, and the CM7_CACR.SIWT bit is set.

The following sequence is required for this erratum to occur:

1. The address of interest must be in the data cache.
2. A Write-Through store is performed to the same double-word as the address of interest.
3. One of the following:
 - A linefill is started (to a different cacheline to the address of interest) that allocates to the same set and way as the address of interest.
 - An *Error Correcting Code* (ECC) error is observed anywhere in the data cache.
 - A data cache maintenance operation without a following *Data Synchronization Barrier* (DSB).

4. A store to the address of interest.
5. A load to the address of interest.

If certain specific timing conditions are met, the load gets the data from the first store, or from what was in the cache at the start of the sequence instead of the data from the second store.

Implications

A load can return incorrect data.

Workaround

There is no direct workaround for this erratum.

Where possible, Arm recommends that you use the MPU to change the attributes on any Write-Through memory to Write-Back memory. If this is not possible, it might be necessary to disable the cache for sections of code that access Write-Through memory.

Category A (rare)

There are no errata in this category.

Category B

412512

Cortex-M7 TPIU might lose trace data in configurations with an ETM but no ITM

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category B
Fault Status: Present in r0p1. Fixed in r0p2.

Description

The Cortex-M7 TPIU outputs ETM and ITM trace data in 16-byte frames as specified by the CoreSight architecture. To complete each frame the TPIU must either wait for sufficient trace data or must receive a DSYNC signal from the DWT. The DSYNC signal is intended to force the TPIU to complete partial frames periodically or whenever the processor halts.

Because of this erratum, the DSYNC signal is inactive, thereby preventing the completion and output of trace data for partial frames.

Configurations affected

This affects configurations with:

- an ETM present AND
- an ITM not present AND
- the Cortex-M7 TPIU.

Conditions

An affected configuration will always suffer from this erratum when the TPIU is enabled.

Implications

This erratum can cause loss of the final bytes of trace data in a tracing session. This applies to a maximum of 15 bytes.

Additionally, when the processor enters halt, it is possible that all the trace packets generated before the halt entry will not be presented to the debug tools when expected. This might cause misinterpretation of the trace data but is not expected to cause decompression errors.

Workaround

There is no workaround for this erratum.

Please note that this erratum is now published as ID 412512. The previous ID 839170 is deprecated. This is done to work around a document generation issue.

426115

Interrupting a FAULTMASK-setting instruction might cause incorrect MPU instruction attributes

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category B
Fault Status: Present in r0p1. Fixed in r0p2.

Description

When the MPU_CTRL.HFNMIENA is clear the MPU uses the default memory map instead of the programmed regions when executing at HardFault or NMI priority. This includes code where FAULTMASK is set.

Because of this erratum, if an instruction attempting to set FAULTMASK is interrupted, all fetch MPU lookups in the interrupt handler might incorrectly be performed at HardFault or NMI priority.

Configurations affected

This erratum affects all configurations of the processor that include an MPU.

Conditions

The following conditions are required for this erratum to occur:

- The MPU is enabled and the MPU_CTRL.HFNMIENA bit is b0.
- The processor is executing at priority >0.
- Execution of a CPS or MSR instruction to set FAULTMASK is attempted.
- An asynchronous exception (that is not NMI) is recognised while this instruction is executing such that it is killed and FAULTMASK is not set.

In this situation, the processor will incorrectly consider all fetch MPU lookups for the interrupt handler to be at HardFault or NMI priority. This effect will continue until either the handler completes and performs a return, or until another exception is taken.

Implications

This erratum only affects instruction fetches.
The MPU attributes returned will be incorrect. This could result in:

- xN faults not taken when they should be
- Spurious xN faults

- Incorrect attributes on the AXI interface that could cause system-specific effects, for example, if system caching is implemented. Note however that since this issue only applies to instruction fetches, system caches are unlikely to return incorrect instruction data in the absence of self-modifying code. If self-modifying code is used, appropriate use of clean and invalidate operations can be used to work around this issue.

Note that this erratum does not represent a privilege violation because the affected code will be still be executed with the correct privilege.

Workaround

The instruction to attempt to set FAULTMASK should only be executed with PRIMASK set.

So the sequence:

<CPS/MSR to set FAULTMASK>

should be replaced by:

CPSID i

<CPS/MSR to set FAULTMASK>

Please note that this erratum is now published as ID 426115. The previous ID 834922 is deprecated. This is done to work around a document generation issue.

440977

Increasing priority using a write to BASEPRI does not take effect immediately

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category B

Fault Status: Present in r0p1. Fixed in r0p2.

Description

An MSR to BASEPRI or BASEPRI_MAX can be used to boost current execution priority.

This update is required to be serialised to the instruction stream meaning that after this update completes, it takes effect immediately and no exceptions of lower priority than the new boosted priority can pre-empt execution.

Because of this erratum, the priority boosting does not take place immediately, allowing the instruction after the MSR to be interrupted by an exception of lower priority than the new boosted priority.

This effect is only limited to the next instruction. Subsequent instructions are guaranteed to see the new boosted priority.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

The following scenario is required to hit this erratum:

1. An MSR to BASEPRI or BASEPRI_MAX is executed to increase current execution priority.
2. An asynchronous exception (interrupt, asynchronous bus fault, SysTick, PendSV) becomes pending around the time that the MSR is being executed. The priority of this exception is higher than the execution priority before the MSR and lower than the priority after it.

Under these conditions, the asynchronous exception might be taken after the MSR completes. In this situation, the return address stacked will point to the instruction after the MSR.

The window for this erratum ends when the instruction after the MSR completes. After this, this erratum cannot occur.

Implications

This erratum means that the instruction after an MSR to boost BASEPRI might incorrectly be pre-empted by an insufficiently high priority exception.

Note that this erratum only affects MSR to BASEPRI or BASEPRI_MAX.
MSR or CPS to PRIMASK or FAULTMASK work correctly.

Workaround

To work around this problem, the MSR to boost BASEPRI can be replaced by the following code sequence:

```
CPSID i  
MSR to BASEPRI  
CPSIE i  
<critical region code>
```

Please note that this erratum is now published as ID 440977. The previous ID 837070 is deprecated. This is done to work around a document generation issue.

565285

Core can send AXI transactions that permit reordering when it should not

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category B

Fault Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open

Description

The AXI protocol allows transactions on different write IDs to be reordered with respect to one another. Because of this erratum, the core might output a write transaction using the ID for cacheable data before the slave asserts BVALID for a write transaction that is the same address and uses the ID for evictions. If the slave or interconnect were to reorder these transactions, then this can result in storing the wrong data in memory. The store data is always output on WDATA in the correct order.

Configurations Affected

All configurations that include a data cache are affected.

Conditions

This erratum requires an eviction followed by a cacheable store to an address that was present in the eviction.

There are two scenarios when this can happen.

In the first scenario, the eviction and write are not directly related.

1. An eviction occurs for any reason
2. A cacheable store, to an address present in the eviction, that causes a write on AXI, not a linefill. This could be caused by any of the following reasons:
 - The store has memory attributes that are Write-Back no Write-Allocate
 - The core has entered dynamic read allocate mode
 - A write-after-write hazard between the store and another AXI write caused the memory system to not start a linefill for that store

In the second scenario, the eviction has been caused by the store.

1. A cacheable store attempts to write to a line that is in the cache and dirty
2. The store looks up in the cache and gets an ECC error, which always (except fatal error on the tag) causes an eviction of a full line, and the store is sent out on AXI regardless of its memory attributes. This could be caused by either:
 - The ECC error being fatal error in the data cache
 - The ECC error being correctable, but both data cache error bank registers are locked

In order for the lookup to get an ECC error, the core must be configured with cache ECC, and it must be enabled.

Implications

If the slave or interconnect does reorder the transactions, the older eviction will overwrite the result of the younger store, causing data corruption.

Workaround

If the core is operating in a system that is susceptible to this erratum, and the core is not configured with cache ECC, or cache ECC is disabled, the following workaround can be used:

1. Avoid the use of Write-Back no Write-Allocate memory programmed in the MPU, *and*
2. Disable Dynamic Read Allocate mode by setting ACTLR.DSRAMODE to 1

Note: If the core is configured without an MPU, or the MPU is disabled, none of the memory is Write-Back no Write-Allocate.

This workaround might have a small performance impact.

If the core is configured with cache ECC, and it is enabled, then the workaround must *also* include:

3. Do not lock both data cache error bank registers

1013783

PLD might perform linefill to address that would generate a MemManage Fault

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Cat B

Fault Status: Present in r0p1, r0p2, r1p0, r1p1, and r1p2. Open.

Description

If the MPU is present and enabled, then it can be programmed so that loads to certain addresses generate a MemManage Fault. This could be because:

- The address is unmapped, that is, it is not in an enabled region and the default memory map is not being used.
- The address cannot be accessed at the current privilege level.
- The address cannot be accessed at any privilege level.

Because of this erratum, a PLD to such an address might incorrectly cause a data cache linefill.

Configurations Affected

This erratum affects all configurations that include an MPU and a data cache.

Conditions

1. The data cache is enabled and the MPU is enabled.
2. A PLD is executed, and either:
 - a. The PLD is to an address not mapped in the MPU, which requires that:
 - i. The MPU is enabled
 - ii. The default memory map is not being used
 - iii. The default memory map is cacheable at that address
 - iv. The PLD does not hit an enabled MPU region.
 - b. The PLD is to a region that has permission requirements that the PLD does not meet, which requires that:
 - i. The MPU is enabled.
 - ii. The default memory map is not being used.
 - iii. The region that the PLD hits is cacheable.
 - iv. The region that the PLD hits would generate a MemManage fault for a load. This requires either:
 - i. The region cannot be accessed by a read at any privilege level.
 - ii. The region only has read access for privileged code and the PLD is unprivileged.

Note that in rare cases, a PLD instruction can be speculatively executed in the shadow of a mispredicted branch. This can even theoretically be a literal value that decodes to a PLD.

Implications

Processor execution is not affected by this erratum. The data returned from the linefill is not directly consumed by the PLD. Any subsequent load to that address can only access the data if it has permissions to do so. This erratum does not permit software to access data that it does not have permissions for.

The only implications of this erratum are the access itself which should not have been performed. This might have an impact on memory regions with side-effects on reads or on memory which never returns a response on the bus.

Workaround

Accesses to memory that is not mapped in the MPU can be avoided by using MPU region 0 to cover all unmapped memory and make this region execute-never and inaccessible. That is, MPU_RASR0 should be programmed with:

- MPU_RASR0.ENABLE = 1 ; MPU region 0 enable
- MPU_RASR0.SIZE = b11111 ; MPU region 0 size = 2^{32} bytes to cover entire memory
- MPU_RASR0.SRD = b00000000 ; All sub-regions enabled
- MPU_RASR0.XN = 1; Execute-never to prevent instruction fetch
- MPU_RASR0.AP = b000; No read or write access for any privilege level
- MPU_RASR0.TEX = b000 ; Attributes = Strongly-ordered
- MPU_RASR0.C = b0 ; Attributes = Strongly-ordered
- MPU_RASR0.B = b0 ; Attributes = Strongly-ordered

Accesses to memory that is mapped in the MPU, but should not be accessed at the current privilege level can be avoided by making the region non-cacheable. That is, MPU_RASR0 should be programmed with:

- MPU_RASR0.TEX = b000 ; Attributes = Strongly-ordered
- MPU_RASR0.C = b0 ; Attributes = Strongly-ordered
- MPU_RASR0.B = b0 ; Attributes = Strongly-ordered

2328489

TCM bandwidth sharing between AHBS writes and software stores might not function correctly when using TCM wait states

Status

Affects: Cortex-M7

Fault Type: Programmer Category B

Fault Status: Present in r0p1, r0p2, r1p0, r1p1, r1p2. Open.

Description

The *TCM Control Unit* (TCU) contains a *Store Queue* (SQ) FIFO that buffers *Tightly Coupled Memory* (TCM) writes. Software stores and AHBS writes both go through the SQ. A round-robin scheme is used to fairly share the SQ ingress bandwidth between software stores and AHBS writes when contention occurs. Due to this erratum, AHBS writes might take priority over 64-bit software stores, which can stall processor instruction execution while the AHBS writes are ongoing. Software stores take priority over AHBS writes and therefore this erratum does not occur when there is adequate TCM bandwidth available.

Configurations affected

This erratum affects all configurations of the Cortex-M7 processor.

Conditions

This erratum can occur when either an STRD, STM, PUSH, VPUSH, VSTR.64, or VSTM instruction is executed targeting TCM during a long stream of back-to-back AHBS write transfers and TCM wait states are used.

Also, if a load instruction closely follows after a 64-bit store instruction to TCM that stalls the execution pipeline, then this will further exacerbate the issue. If the pipeline is stalled, the read will still go ahead but the data returned will be discarded. Hence, a TCM read will be repeated until the stall ends.

Implications

This erratum only affects the performance of the Cortex-M7 processor. It can cause the processor's execution pipeline to stall when executing a 64-bit store instruction while back-to-back AHBS writes are ongoing. When the AHBS back-to-back write stream ends, the 64-bit store will be accepted into the SQ and the pipeline will stop stalling.

Workaround

Either one of the following workarounds can be used:

1. Use AHBS priority demotion by setting the CM7_AHBSCR register CTL field to 0b00 and set the INIT field value to N.
2. Ensure that there are N consecutive BUSY or IDLE transfers on the AHBS interface for every sequence of 16 SEQ or NONSEQ back-to-back write transfers.

Where, $N \geq (T/2) + 1$ and T is the maximum number of cycles it takes to perform a sequence of two read and one write transaction on a TCM interface.

Category B (rare)

443753

A sequence of cacheable stores to memory locations that always return bus faults might cause deadlock

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category B (rare)
Fault Status: Present in r0p1. Fixed in r0p2.

Description

Because of this erratum, after a specific sequence of cacheable stores, the processor might deadlock.

Configurations affected

This erratum affects all configurations of the processor that include a D-cache.

Conditions

The following code sequence is required to hit this erratum:

1. A cacheable WBWA store, that misses in the cache.
2. At least one of the following:
 - A LDREX, STREX, DSB, or AHBD load that is interrupted by an exception.
 - The MPU is reprogrammed, and there is no DSB between that and the next store.
3. Three cacheable WBWA stores, all of which must:
 - Be to locations such that the linefill that is triggered will always get a bus fault.
 - Not get a memory management fault.

After this sequence, provided certain timing specific conditions are met, there will be four stores that will never drain from the STB. If the processor attempts to execute another store to AXI, DSB or DMB, the processor will deadlock.

Implications

This erratum results in the processor deadlocking.

An exception will break the deadlock, but cannot prevent the processor from deadlocking again if another store to AXI, DSB or DMB, is executed. If the stack is in AXI, the processor will deadlock when it attempts to perform stacking.

When in this deadlock state, the processor will not be able to enter halt state or service debugger accesses to memory.

Workaround

Systems that do not implement permanently bus-aborting regions of the AXI interface are not affected by this erratum and need no workaround.

For systems that are affected, the recommended workaround is for SW to program the MPU such that no accesses to these permanently aborting regions are allowed.

If this workaround is not possible, then the CACR.FORCEWT bit can be set to force all cacheable stores to be treated as write-through. Note that this might cause some performance degradation.

Code should avoid accesses to areas generating bus faults.

Please note that this erratum is now published as ID 443753. The previous ID 838169 is deprecated. This is done to work around a document generation issue.

Category C

399743

The Fault Address Register (FAR) might be corrupted when BFHFNMIGN is set

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category C
Fault Status: Present in r0p1. Fixed in r0p2.

Description

Cortex-M7 implements a single physical register (FAR) for both the Bus Fault Address Register (BFAR) and the MemManage Fault Address Register (MMFAR).

Because of this erratum this register might become corrupted in rare cases.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

There are two sets of conditions for this erratum.

The first set is:

1. The CCR.BFHFNMIGN bit is set.
2. The processor is executing at HardFault or NMI priority - that is, either FAULTMASK is set or the processor is executing the HardFault or NMI handlers.
3. A load-to-PC instruction is executed that is subject to a bus fault.

In this situation the value in the FAR should be the load address used by the faulting load-to-PC instruction. Because of this erratum the FAR might incorrectly be updated with a random value.

The second set is:

1. The CCR.BFHFNMIGN bit is set.
2. The processor is executing in unprivileged Thread mode with FAULTMASK set.
3. A load is executed that is subject to a bus fault. This load can be to any destination register.
4. The instruction following the load is executed on the same cycle and is a VFP load/store instruction that is subject to all of the following conditions:
 - a. Causes a NOCP UsageFault exception to be raised because the CPACR is configured to enable privileged access only to the FPU. Note that if the CPACR is configured in any other way,

- including all access disabled, then this failure mode cannot occur.
- b. The MPU is set up such that the load/store address defined by the instruction is not allowed access by unprivileged requests.

In this situation the value in the FAR should be the load address used by the first load instruction. Because of this erratum however, the FAR might incorrectly be updated with a random value.

Implications

Corruption of the FAR can cause confusion in the fault handling code as to what the source of the fault was, but is not otherwise expected to affect SW operation.

Additionally, the CCR.BFHFNMIGN bit is expected to be set in a very restricted set of scenarios, which are unlikely to include:

1. A load-to-PC subject to a bus fault. Note that this would, even without this erratum, potentially result in unpredictable program flow because the PC is being written with potentially random data on the bus.
2. Execution in unprivileged thread mode.

Therefore it is expected that this erratum will not affect the majority of systems.

Workaround

To work around this erratum, SW should ensure that the CCR.BFHFNMIGN bit is only set when executing in privileged mode and should avoid executing load-to-PC instructions to addresses that could potentially fault when CCR.BFHFNMIGN is set.

Please note that this erratum is now published as ID 399743. The previous ID 830969 is deprecated. This is done to work around a document generation issue.

408519

Incorrect GTS packet generation when global timestamps are enabled during debug using the ITM

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category C
Fault Status: Present in r0p1. Fixed in r0p2.

Description

A full global timestamp packet sequence should be generated whenever the GTS feature is enabled in the ITM. This packet sequence consists of a GTS1 packet followed by a GTS2 packet.

Because of this erratum, enabling the GTS feature in the ITM can cause GTS1 packets to be continuously output without a subsequent GTS2 packet.

Configurations affected

This erratum affects all configurations of the processor that include an ITM.

Conditions

The following condition is required to hit this erratum:

- A write to the ITM_TCR register setting the GTSFREQ bits to a non-zero value.

This should cause a single request for a full GTS packet to be output. Because of this erratum however, the request for a GTS packet is incorrectly held high until the next write to the PPB space of the memory map.

This causes GTS1 packets to be continuously output and prevents the output of the required GTS2 packet.

Implications

This erratum results in wastage of ITM trace bandwidth because of the unnecessary GTS1 packets output, together with the loss of global timestamp information because of the failure to output a GTS2 packet.

Workaround

To work around this erratum, a dummy write to the ITM_TCR register should be performed immediately after the GTS feature is enabled. The simplest way to do this is to perform the write to enable the GTS feature twice, back-to-back.

This workaround is required for both SW and debugger writes.

Please note that this erratum is now published as ID 408519. The previous ID 839169 is deprecated. This is done to work around a document generation issue.

416915

HFSR.FORCED bit is not set for configurable priority faults which result in LOCKUP

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category C
Fault Status: Present in r0p1. Fixed in r0p2.

Description

Configurable priority faults can escalate to HardFault when they are not high enough priority to preempt the current executing context. In this situation, they should set the HFSR.FORCED bit to indicate that they have been escalated to HardFault. This should also apply when these faults occur at NMI or HardFault priority, thus resulting in LOCKUP.

Because of this erratum, the HFSR.FORCED bit is not set when configurable-priority faults cause LOCKUP.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

The following conditions are required to hit this erratum.

1. The processor should be executing at HardFault or NMI priority.
2. Any of the following faults are raised:
 - MemManage
 - Usage
 - Synchronous BusFault - that is, a BusFault on an instruction fetch or load.

In these situations, the processor fails to set the HFSR.FORCED bit but in every other respect, correctly handles the fault by:

1. Setting the appropriate FSR bit for the original fault exception.
2. Entering LOCKUP.

Implications

This erratum only affects the value of the HFSR.FORCED bit in an unrecoverable scenario and has no other effects on processor operation.

This bit has no internal function in the processor and is only used to help handler SW or a debugger to understand the source of LOCKUP. Therefore, this erratum can make debug of a faulty system more difficult.

Workaround

There is no workaround for this erratum.

In general, it is not anticipated that the conditions and sequence described will occur in real code.

Please note that this erratum is now published as ID 416915. The previous ID 834971 is deprecated. This is done to work around a document generation issue.

421025

Early forwarding from load is incorrectly cancelled inside IT block

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category C
Fault Status: Present in r0p1. Fixed in r0p2.

Description

Because of this erratum, a specific sequence of instructions inside an IT block might cause an incorrect result to be computed in the integer registers.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

The following code sequence is required to hit this erratum:

1. ITTTE with the first three instructions passing their condition code checks.
2. A word-aligned single word load to register Rd.
3. A word aligned single word load to register Rd with no operand dependency on the previous load.
4. A flag setting data processing instruction with a destination register different to Rd.
5. A data processing operation that:
 - Passes its condition code check only because of the flag setting operation in step 4 changing the flags.
 - Uses Rd as a source operand.

Under these and other code-alignment and timing specific conditions, the last data processing instruction might incorrectly use the data from the first load instead of the second and therefore generate an incorrect result.

Implications

This erratum results in data corruption of the integer registers.

Note that this code sequence is not expected to be generated by C compilers because the first load in the code sequence is completely redundant.

ARM C Compiler 5.03 , gcc 4.9-2015-q1-update (and gcc 5.0 trunk) and IAR EWARM 7.40 have been verified as not generating this code sequence.

Workaround

The code sequence must be amended to remove the redundant load operation.

Please note that this erratum is now published as ID 421025. The previous ID 833872 is deprecated. This is done to work around a document generation issue.

422825

MPU fetch attributes might transiently be incorrect after an exception return

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in r0p1. Fixed in r0p2.

Description

When the MPU_CTRL.HFNMIENA is clear, the MPU uses the default memory map instead of the programmed regions when executing at HardFault or NMI priority.

Because of this erratum, this setting might cause instruction fetches to return incorrect attributes from the MPU.

Configurations affected

This erratum affects all configurations of the processor that include an MPU.

Conditions

The following conditions are required for this erratum to occur:

- The MPU is enabled and the MPU_CTRL.HFNMIENA bit is b0.
- An exception return is executed in a handler at HardFault or NMI priority.
- The exception return target is not at HardFault priority.
- During the exception return, the processor's fetch unit is stalled sufficiently to prevent the first fetch from the exception return target from being performed at the correct priority.
 - If the ITCM is implemented and enabled, at least 4 cycles of wait states on the ITCM interface are required to hit this erratum. Alternatively, either the MBIST or AHBS interface should be accessing the ITCM around the time the exception return executes.
 - If the ITCM is not implemented or not enabled, the MBIST interface should be accessing the I-cache near the time the exception return executes.

While the processor is running, if the MBIST interface is not in use and there is no AHBS traffic to the ITCM, then this set of conditions cannot occur provided the ITCM interface is guaranteed to insert no more than 3 cycles of wait-states.

In these scenarios, the MPU lookup for the first few instructions at the exception return target might incorrectly be performed at HardFault or NMI priority and therefore return incorrect MPU attributes.

Note that this erratum only affects the MPU lookups and does not affect other aspects of execution.

Implications

If the affected instructions are on ITCM, then all MPU attributes are ignored except for the xN attribute. If the affected instructions are on AXI, then the attributes on the AXI interface might be incorrect, causing other effects in the system, if for example, system caches are implemented.

If the xN attribute is returned incorrectly, the instructions at the target might either be subject to spurious faults, or they might execute instead of faulting. Note that if they do execute, they will do so at the correct privilege level and therefore cannot themselves circumvent their privilege setting in the processor.

Workaround

No workaround is required for this erratum.

Please note that this erratum is now published as ID 422825. The previous ID 834923 is deprecated. This is done to work around a document generation issue.

423541

Interrupts on a bus-aborting strongly-ordered or device load to the stack pointer might cause incorrect exception stacking

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category C
Fault Status: Present in r0p1. Fixed in r0p2.

Description

A load-single instruction subject to a bus-abort should leave the processor in a state that allows the load to be re-executed on return from the bus-abort handler.

Because of this erratum, if another asynchronous exception is recognized on such a load instruction, the state of the processor might be incorrect.

Configurations affected

The erratum affects all configurations of the processor that include an FPU.

Conditions

The following scenario is required to hit this erratum:

- A data processing VFP instruction is dual-issued with a load-single instruction.
 - The load instruction needs to be immediately after the VFP instruction in the code image.
- The load is to Strongly-ordered or Device memory.
- The load must update the currently active stack pointer.
 - This can either be by directly loading the stack pointer or by using the stack pointer as the base register and using base write-back.
- The load is subject to a bus-abort on either the AHBP, AXI, TCM, or EPPB interface.
 - The internal PPB address space is not subject to this erratum.
- A separate asynchronous exception (external interrupt, NMI, SysTick, asynchronous bus-abort, and PendSV) is recognized while the load is executing.

Under these and other specific timing conditions, the automatic exception entry stacking might be performed to the wrong address. This includes the final update made to the stack pointer and, if lazy stacking is enabled, the update made to the FPCAR.

Implications

This erratum might result in data corruption in memory and in the integer register file.

Note that loads to Device and Strongly-ordered memory are unlikely to update the stack pointer in real code. Loads that do not update the stack pointer are not subject to this erratum.

Workaround

This erratum is not considered to need a workaround. Code should avoid this scenario.

Please note that this erratum is now published as ID 423541. The previous ID 834924 is deprecated. This is done to work around a document generation issue.

431216

Unimplemented bits of BASEPRI do not read-as-zero

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in r0p1. Fixed in r0p2.

Description

The number of implemented bits of BASEPRI is configurable with all unimplemented bits defined as read-as-zero.

Because of this erratum, unimplemented bits do not read as zero and instead read with the value written to them. This applies to both BASEPRI and its alias BASEPRI_MAX.

Note that this erratum is limited to the read value of BASEPRI and BASEPRI_MAX. For all other aspects of operation including exception priority processing, the unimplemented bits of BASEPRI are correctly treated as zero.

Configurations affected

This erratum affects all configurations of the processor where the number of supported exception priority bits is less than 8.

Conditions

The following sequence is required to hit this erratum:

1. A write to BASEPRI or BASEPRI_MAX, either by debugger or MSR instruction.
 - A 1 must be written to any of the unimplemented, lower order bits.
2. A read of BASEPRI or BASEPRI_MAX, either by debugger or MRS instruction.

Implications

The read value of BASEPRI and BASEPRI_MAX is incorrect and software using either of these registers to identify how many priority bits are implemented will get the wrong result.

There are no other implications for this erratum. All other aspects of BASEPRI operation correctly treat the unimplemented bits as zero.

Workaround

SW should use another priority register to deduce how many priority bits are implemented and should mask unimplemented bits on BASEPRI and BASEPRI_MAX reads.

Please note that this erratum is now published as ID 431216. The previous ID 837069 is deprecated. This is done to work around a document generation issue.

449383

Write to FPCCR.ASPEN while a Single-precision FP MAC is completing might corrupt the FP register bank

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category C
Fault Status: Present in r0p1. Fixed in r0p2.

Description

Single-precision floating-point multiply-accumulates (MACs) require a number of cycles to execute. If during this time, the FPCCR.ASPEN is changed to enable automatic FP state preservation, and the default rounding controls differ from the current settings, a subsequent FP instruction might cause the wrong value to be written to the FP register file.

Configurations affected

This erratum affects all configurations of the processor that include the floating-point unit (FPU).

Conditions

The following sequence is required to hit this erratum:

1. The processor is not currently in automatic FP state-preservation mode and the default and current FP rounding, NaN or flush-to-zero modes differ.
 - CONTROL.FPCA = 0, FPCCR.ASPEN = 0, and FPDSCR differs from FPSCR.
2. A single-precision FP MAC (VMLA, VMLS, VNMLA, VNMLS, VFMA, VFNMS, VFNMA, VFNMS).
3. Within the next eight instructions, and before any exception occurs, there is:
 - A software write that sets the FPCCR.ASPEN bit.
 - Any subsequent FP instruction.

This erratum can occur even if there is a single DSB between the FPCCR.ASPEN write and the next FP instruction. But in this case, the MAC, the write to FPCCR, and the DSB must occur back-to-back.

Implications

After this sequence, provided certain timing specific conditions are met, the result of either of the FP instructions might be incorrect.

Workaround

In general, it is not anticipated that the conditions and sequence described will occur in real code. In the case where a workaround is required, ensuring the FPCCR write is followed by a DSB, ISB sequence before executing the subsequent FP instruction will avoid this erratum.

Please note that this erratum is now published as ID 449383. The previous ID 839269 is deprecated. This is done to work around a document generation issue.

486321

Incorrect behavior of profiling counters

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Category C
Fault Status: Present in r0p1, r0p2 and r1p0. Fixed in r1p1.

Description

The profiling counters are provided to enable non-intrusive counting of events with limited accuracy. Due to this erratum, some events update the wrong event counter.

The DWT_LSUCNT counter should be incremented for all extra cycles that the processor spends executing normal load-store instructions. Because of this erratum, stall cycles in the memory system will cause the DWT_CPICNT to be incremented instead of the DWT_LSUCNT. The DWT_LSUCNT will only count non-stalled extra cycles spent handling load-store instructions.

The DWT_EXCCNT counter should be incremented whilst lazy VFP state preservation is being performed. Because of this erratum, these cycles increment the DWT_LSUCNT counter and, on stall cycles, the DWT_CPICNT instead.

Note that no cycles are missed entirely or double counted. This erratum only means that the cycles are counted in the wrong counter.

Configurations affected

This erratum affects all configurations of the processor.

Conditions

The following conditions are required to hit this erratum:

- The DEMCR.TRCENA bit is set
- Any or all of the following bits are set:
 - DWT_CTRL.LSUEVTENA to enable the DWT_LSUCNT counter
 - DWT_CTRL.CPICNT to enable the DWT_CPICNT counter
 - DWT_CTRL.EXCCNT to enable the DWT_EXCCNT counter

Implications

This erratum results in incorrect values in the profiling counters.
It has no other functional impact.

Workaround

There is no workaround for this erratum.

Please note that this erratum is now published as ID 486321. The previous ID 850724 is deprecated. This is done to work around a document generation issue.

505438

TPIU cannot be flushed in Debug state if Cortex-M7 TPIU is used

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in r0p2 and r1p0. Fixed in r1p1.

Description

The Cortex-M7 TPIU requires a DSYNC to flush any trace data that does not form a full frame and to allow trace synchronization.

A write to the DWT_CYCCNT register in the DWT is specified to generate this DSYNC to the Cortex-M7 TPIU.

This mechanism provides a means to flush out any buffered trace data in the TPIU.

Because of this erratum, a write to the DWT_CYCCNT register does not generate this DSYNC.

Configurations affected

This affects configurations with:

- An ITM present, or an ETM present, or both present, and
- The Cortex-M7 TPIU.

Conditions

An affected configuration will always suffer from this erratum when the TPIU is enabled.

Implications

This erratum can cause loss of the final bytes in a trace session. This applies to a maximum of 14 bytes. Additionally, it leaves an external debugger no way to flush out any buffered trace data.

This affects the standard step and trace model such that data generated during a step event will not be visible until a subsequent entry into halt mode.

Workaround

The DSYNC signal can be manually triggered, under the same conditions as the disabled method, by causing the processor to leave halted state and then halting the processor a second time.

Please note that this erratum is now published as ID 505438. The previous ID 850725 is deprecated. This is done to work around a document generation issue.

513195

Lock Status Indication incorrectly reads as one for debugger reads

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in: r0p1, r0p2 and r1p0. Fixed in r1p1.

Description

The FPB, DWT, and ITM components each implement a CoreSight lock mechanism.

This lock is intended to prevent accidental software writes to control registers and is not required for external debugger accesses.

For an external debugger:

- LSR.LSI should RAZ to show that the lock is not implemented.
- LSR.SLK should RAZ to show that the component is not locked.

Because of this erratum, the LSR.LSI bit incorrectly reads as one for debugger accesses.

Configurations affected

This erratum affects all configurations of Cortex-M7.

Conditions

An external debugger read to any of the following registers will observe this erratum:

- FPB_LSR
- DWT_LSR
- ITM_LSR

Implications

An external debugger reading the LSR might be misled by the incorrect LSR.LSI value and attempt to obtain the lock. If the debugger polls until LSR.SLK reads as one, then the debugger could livelock. If the debugger does not poll on LSR.SLK, and instead assumes it has the lock and carries on, then there are no implications of this erratum.

Workaround

External debuggers should ignore the SLI field of the FPB_LSR, DWT_LSR and ITM_LSR registers. External tools should not attempt to lock or unlock the lock by writing to FPB_LAR, DWT_LAR and ITM_LAR registers.

Please note that this erratum is now published as ID 513195. The previous ID 851031 is deprecated. This is done to work around a document generation issue.

636315

Software programming errors might not be reported for on-line MBIST access to the I-Cache

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Cat C

Fault Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open.

Description

The on-line MBIST interface provides access to the cache and TCM RAMs to allow in-field memory testing during normal operation of the processor. Because of this erratum, errors in the software that works in conjunction with the memory testing might not be indicated on the MBISTERR output signal as intended for I-Cache tests.

Note that this erratum does not affect the detection of faults in the memories under test, but affects only the feature that helps to indicate errors in software used during testing.

There are two on-line MBIST use cases: software transparent and software assisted.

In the software transparent use case, software running on the processor is not involved in or aware of the memory testing being carried out. See the Cortex-M7 Safety Manual for more details. In this case the target memory is automatically locked by the MBIST controller, which causes the processor pipeline to stall if it attempts to access this memory. Testing is carried out using short bursts of accesses which last for less than 20 clock cycles and do not corrupt the memory contents. For this reason the memory is locked only for a very short period of time and the gap between bursts is very large.

In the software assisted use case the target memory is still locked by the MBIST controller but software running on the processor disables the target memory before testing commences. This prevents any software access to this memory during testing. See the Cortex-M7 Safety Manual for more details. For this reason, software accesses will go to another memory instead of the target memory and the pipeline will not stall. This is important because the software assisted use case is intended to be used for production MBIST algorithms, which take a long time to run. For example, if the I-Cache were disabled then software might still execute using the main memory or the TCMs.

This erratum only affects the software assisted use case, when the I-Cache RAMs are tested. An error indication is sent back to the MBIST controller if software attempts to access the target memory while it is locked for testing. Because of this erratum, an error is not indicated back to the MBIST controller on the MBISTERR[0] output signal when software performs a lookup to the I-Cache during MBIST testing. The error indication is correctly asserted for all the other type of I-Cache access during MBIST testing:

- A cache line invalidate because of ECC error.
- A cache invalidate by MVA.
- A cache invalidate all operation.
- A cache line-fill allocation.

Note that this erratum only affects the MBIST software assisted use case error indication for the I-Cache and the MBISTERR[0] signal functions correctly for the D-Cache, ITCM and DTCM.

Configurations affected

This erratum affects configurations of the processor that include I-Cache RAMs.

Conditions

The following conditions are required to cause this erratum:

- The software intends to use the software assisted on-line MBIST use case.
- The I-Cache is not disabled by software running on the Cortex-M7 before testing commences.
- The MBIST controller selects an I-Cache memory array for testing, locks the target memory and testing commences.
- The software running on the Cortex-M7 causes an instruction fetch from the I-Cache when it expected to fetch from main memory.

Implications

This erratum could result in an error not being indicated back to the MBIST controller on the MBISTERR[0] output signal when software assisted use case is used and the I-Cache is not disabled by software before testing commences. This could result in the processor unexpectedly stalling for a long period of time during MBIST testing of the I-Cache memories, without there being a clear indication of the cause of the stall. For this reason, the processor might not make progress as expected, because of the software error, during I-Cache testing.

Workaround

There is no workaround for this erratum.

702596

Single stepping Cortex-M7 enters pending exception handler

Status

Affects: Cortex-M7, Cortex-M7 with FPU
Fault Type: Programmer Cat C
Fault Status: Present in r0p1. Fixed in r0p2.

Description

Setting the DHCSR.C_MASKINTS bit to 1 by the debugger should prevent the interrupts from being taken. This functionality can be used by the debugger to prevent the processor from entering an interrupt handler while single stepping.

As the result of this erratum an interrupt that should be masked by DHCSR.C_MASKINTS can be taken when the processor leaves Debug state.

Configurations affected

All configurations are affected.

Conditions

- An enabled interrupt is either pending when the processor enters Debug state, or becomes pending while the processor is in Debug state. This includes PendSV, SysTick and external configurable interrupts.
- At some point while the processor is in Debug state and the interrupt is pending, DHCSR.C_MASKINTS is zero. This might be because either DHCSR.C_MASKINTS was zero when the processor entered Debug state, or the debugger has cleared DHCSR.C_MASKINTS after the processor entered Debug state.
- The processor exits Debug state.

Implications

Because of this erratum the debugger will enter the exception handler of a pending interrupt while single stepping through the code regardless of the value of DHCSR.C_MASKINTS. New interrupts will not be invoked while the C_MASKINTS bit is kept high.

Workaround

There is no complete workaround for this erratum.

The debugger can reduce the likelihood of an interrupt being taken when `DHCSR.C_MASKINTS` is set by:

- On detecting that the processor has entered the halted state:
- Setting `DHCSR.C_MASKINTS` as soon as possible, if `DHCSR.C_MASKINTS` was previously clear.
- Avoid clearing `DHCSR.C_MASKINTS`.
- Subsequently only clearing `DHCSR.C_MASKINTS` when leaving Debug state if the user requests execution with `DHCSR.C_MASKINTS` clear.

If the user performs a sequence of steps with `DHCSR.C_MASKINTS` set following these rules, then only the window between the processor first entering Debug state and the debugger first setting `DHCSR.C_MASKINTS` will be prone to this erratum.

1267980

ECC error causes data corruption when the data cache error bank registers are locked

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open.

Description

The data cache contains two error bank registers, DEBR0 and DEBR1. These registers store the locations in the cache that *Error Correcting Code* (ECC) errors affect and prevent future allocations to those locations.

Software can lock each DEBR and this prevents the DEBR from being automatically updated when a data cache ECC error is detected.

Because of this erratum, if both DEBR0 and DEBR1 are locked and an ECC error is detected on a cacheable store, then the store data is written onto the bus but not written into the data cache. This might result in the data cache containing stale data.

Configurations Affected

All configurations with a data cache and ECC are affected.

Conditions

- DEBR0 and DEBR1 are locked.
- The wanted address has been allocated to the cache.
- A cacheable store to the wanted address looks up in the cache, and an ECC error is found in the cache set that the store addresses.

Implications

This erratum can cause data corruption in the data cache.

Workaround

Software must avoid locking both error bank registers.

1313001

Store after cache invalidate without intervening barrier might cause inconsistent memory view

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open.

Description

If a cache invalidate operation is followed by a Write-Through store to an address affected by that operation and a linefill to that address occurs, then the linefill might allocate to the cache without the data from the store. Subsequently, that store writes to the bus and leaves the cache with stale data.

Configurations Affected

All configurations with a data cache and without *Error Correcting Code* (ECC) are affected.

Conditions

The following sequence is required for this erratum to occur:

1. The address of interest is in the cache.
2. One of the following data cache maintenance operations that affects the same cache line as the wanted address is performed.
 - DCCIMVAC.
 - DCCISW.
 - DCIMVAC.
 - DCISW.
3. A Write-Through store is performed to the wanted address
4. A linefill to the same cache line of the wanted address occurs for any reason.

There must be no DSB or DMB between the maintenance operation and the store.

If this sequence occurs and certain very specific internal timing conditions are met, then the store data is not merged into the linefill, but it writes out to the bus. After this has occurred, the linefill buffer or cache contains stale data.

Implications

A subsequent load to the same address of the store might observe stale data in the cache.

Workaround

A DMB must be inserted between the cache maintenance operation and the store.

It is expected that all code should already have this DMB or DSB because there is no implicit ordering between cache maintenance operations and stores.

1315869

Data corruption for load following Store-Exclusive

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open.

Description

A load that follows a Store-Exclusive to the same address might forward data from an earlier store, situated between the Load-Exclusive and the Store-Exclusive, and not the data from the Store-Exclusive.

Configurations Affected

All configurations are affected.

Conditions

The following sequence is required for this erratum to occur:

1. A load exclusive sets the local monitor.
2. A store to the wanted address
3. Any of the following instructions to the wanted address. This instruction must not fail either the local or global monitor check.
 - STREXB.
 - STREXH.
 - STREX.
4. A load to the wanted address.

There must be at most one instruction between the Store-Exclusive and the load. All accesses must be to Shareable memory.

Implications

Data corruption occurs when the load returns data from the older store instead of the newer Store-Exclusive.

Stores between a Load-Exclusive and Store-Exclusive are not expected in real code because such stores can always clear the local monitor in some implementations.

Workaround

No workaround is necessary.

1518990

Value used for DWT Data Value Comparison is in memory-endianness format, not little-endian

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault: Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open.

Description

The DWT comparators can be configured to match, and generate a trace event, when data is loaded or stored, and the value of the data matches the value programmed into the DWT_COMPn registers.

Due to this erratum, when in a big-endian system, the data value in DWT_COMPn is considered to be in big-endian format, not little-endian.

Configurations affected

Only big-endian configurations with a DWT are affected.

Conditions

A DWT comparator must be:

- Enabled and either generating a trace packet or triggering a watchpoint (DWTFUNCTIONn.FUNCTION is 0b0101, 0b0110, 0b0111, 0b1001, 0b1010 or 0b1011)
- Performing a data value comparison (DWTFUNCTIONn.DATAVMATCH is 0b1)
- The size of the comparison is halfword or word (DWTFUNCTIONn.DATAVMATCH is 0b01 or 0b10)

The core must perform a memory access that the DWT comparator is sensitive to.

Implications

The DWT might not generate a match when it should, or generate a match when it should not. This might result in incorrect data trace, or incorrect watchpoint generation.

Workaround

In a big-endian system, the value programmed into DWT_COMPn should be in the big-endian format, that is the order of bytes in each chunk should be swapped, with the size of each chunk given by DWT_FUNCTIONn.DATAVSIZE.

3092511

Cortex-M7 can halt in an incorrect address when breakpoint and exception occurs simultaneously

Status

Affects: Cortex-M7, Cortex-M7 with FPU

Fault Type: Programmer Category C

Fault Status: Present in r0p1, r0p2, r1p0, r1p1 and r1p2. Open

Description

When an asynchronous exception occurs at the same time as a breakpoint event (either hardware breakpoint or software breakpoint), it is possible for the processor to halt at the beginning of the exception handler instead of the instruction address pointed by the breakpoint.

Configurations Affected

This erratum affects all configurations of Cortex-M7.

When this happens:

- The BKPT bit in *Debug Fault Status Register* (DFSR) is set, indicating that a breakpoint event has occurred.
- The return address of the exception is the breakpoint address. As a result, if the debugger clears the halting control bit in the processor at this point, the processor will reach the breakpoint again after servicing the exception.

The correct behavior should be one of the followings:

1. Execute BKPT instruction and halt at BKPT before taking the asynchronous exception
2. Take the asynchronous exception before BKPT and return to BKPT instruction and then halt on BKPT instruction.

In both cases, the debugger should see the processor halt on the BKPT instruction.

Conditions:

The following conditions are required:

1. Halt mode debug is enabled and is permitted by debug authentication configuration
2. The processor reaches the breakpoint at the same time that the asynchronous exception is invoked

Implications

A debugger connected to the Cortex-M7 can detect the processor is halted after a breakpoint is hit, but might not be able to determine which breakpoint has triggered the halting.

Workaround:

This issue only affect the debugger's operation. The debugger could report the halting reason as an unknown breakpoint, and optionally resume operation. If the processor's operation is resumed, it is likely to be halted again immediately after the interrupt is serviced and returns to the breakpoint address.

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 [0x0y] 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.