Arm Processor

Cortex^{TM}-M3 (AT420) and Cortex-M3 with ETM (AT425)

Product Revision r1p1, r2p0, r2p1

Software Developers Errata Notice

Non-Confidential - Released

Web address

http://www.arm.com/

Feedback on the product

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

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

Feedback on this document

If you have any comments about this document, please send an email to errata@arm.com giving:

- The document title
- The document number, ARM-ECM-0426803
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments

Arm also welcomes general suggestions for additions and improvements.

Release Information

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 in Chapter 2. Fixed errata are not shown as updated unless the erratum text has changed. The summary table in section 2.2 identifies errata that have been fixed in each product revision.

11 Jan 2019: Changes in Document v3

Page	Status	ID	Cat	Rare	Summary of Erratum
13	New	1320266	CatC		Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address

02 Dec 2014: Changes in Document v2

Page	Status	ID	Cat	Rare	Summary of Erratum
16	New	806421	CatB		ITM can deadlock when global timestamping enabled
16	New	838469	CatB	Rare	Store immediate overlapping exception return operation might vector to incorrect interrupt

20 Mar 2013: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
15	New	789571	CatB		Flash patch control register value might be incorrect

Contents

Chapter 1. **Introduction**

This chapter introduces the errata notice for the processors Arm Cortex-M3 and Cortex-M3 with ETM.

1.1. Scope of this document

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

- the current status of the defect
- where the implementation deviates from the specification and the conditions under which erroneous behavior occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a 'work-around' where possible

This document describes errata that may impact anyone who is developing software that will run on implementations of this Arm product.

1.2. Categorization of errata

Errata recorded in this document are split into the following levels of severity:

Table 1 Categorization of errata

Errata Type	Definition
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.

Chapter 2.
Errata Descriptions

2.1. Product Revision Status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n*	Identifies the major revision of the product.
p*n*	Identifies the minor revision or modification status of the product.

2.2. Revisions Affected

Table 2 below lists the product revisions affected by each erratum. A cell marked with $\boxed{X}$ indicates that the erratum affects the revision shown at the top of that column.

This document includes errata that affect revision **Error! Reference source not found.** only.

Refer to the reference material supplied with your product to identify the revision of the IP.

Table 2 Revisions Affected

ID	Cat	Rare	Summary of Erratum	r1p1	r2p0	r2p1
806421	CatB		ITM can deadlock when global timestamping enabled			X
789571	CatB		Flash patch control register value might be incorrect		X	
752419	CatB		Interrupted loads to SP can cause erroneous behaviour	X	X	X
740455	CatB		SVC and BusFault/MemManage may occur out of order	X	X	
641267	CatB		Bit-band access could read or write wrong bit in BE8	X	X	
602117	CatB		LDRD with base in list may result in incorrect base register when interrupted or faulted	X	X	
563915	CatB		Event Register is not set by interrupts and debug	X	X	
838469	CatB	Rare	Store immediate overlapping exception return operation might vector to incorrect interrupt	X	X	X
674118	CatC		TBH will never cause an alignment fault	X	X	
661722	CatC		External event might be lost when core is sleeping	X	X	
616516	CatC		Incorrect core feature identification registers	X	X	
607266	CatC		ETM traces BKPT as an executed instruction	X	X	
548721	CatC		Internal write buffer could be active whilst asleep	X		
532314	CatC		DWT CPI counter increments during sleep	X		
511864	CatC		Cortex-M3 may fetch instructions using incorrect privilege on return from an exception	X		
463769	CatC		Unaligned MPU fault during a write may cause the wrong data to be written to a successful first access	X		
463764	CatC		Core may freeze for SLEEPONEXIT single instruction ISR	X		

ID	Cat	Rare	Summary of Erratum	r1p1	r2p0	r2p1
463763	CatC		BKPT in debug monitor mode can cause DFSR mismatch	X		
1320266	CatC		Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address	X	X	X

2.3. Category A

There are no errata in this category

2.4. Category A (Rare)

There are no errata in this category

2.5. Category B

563915: Event Register is not set by interrupts and debug

Category B
Products Affected: Cortex-M3, Cortex-M3 with ETM.
Present in: r1p1, r2p0

Description

The event register used for WFE wake-up events should be set for the following conditions:

1) Event communication (including SEV on local processor)
2) Any exception entering pending state when SEVONPEND is set
3) Exception entry
4) Exception exit
5) Debug event when debug is enabled

In r0p0, r1p0, r1p1 and r2p0 versions of Cortex-M3 the event register is not set for the exception entry, exception exit or debug events.

Conditions

1) An interrupt or debug event occurs whilst the internal event register is clear and the core is not sleeping
2) A WFE is executed
3) No further interrupts or events occur

Implications

If interrupts related to a WFE sleep can be generated before the WFE is executed then it may be possible for the event to be missed. The interrupt will occur and the handler will be executed for that interrupt but the event register will not be set. When the WFE is executed it will go to sleep and not wake up if no other events or interrupts occur.

Workaround

An implementation time workaround is to connect up the missing events to the RXEV input of Cortex-M3. This can be achieved by decoding interrupt events using ETMINTSTAT as well as using the HALTED output. RXEV needs to be asserted whenever ETMINTSTAT is equal to 3'b001 or 3'b010 or when HALTED is asserted. Since ETMINTSTAT is part of the ETM interface the ETM interface needs to be enabled for this workaround. This is achieved by asserting the ETMPWRUP input on Cortex-M3.

A software workaround is to insert the SEV instruction at the beginning and end of all exception handlers.

602117: LDRD with base in list may result in incorrect base register when interrupted or faulted

Category B

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0

Description

LDRD with the base register in the list of the form `LDRD Ra, Rb, [Ra, #imm]` may not complete after the load of the first destination register due to an interrupt before the completion of the second load or due to the second load getting a bus fault or an MPU fault. Since the base register has been updated the base register must be restored to its original value before entering the appropriate interrupt or fault handler so that the instruction can restart correctly upon return from the handler. In certain circumstances this may not occur as required.

When the LDRD is interrupted in between the two loads then the base register may not be restored as required. This can only happen when the instructions are being executed from the system bus (address 0x20000000 and above) and the loaded data is also being read from the system bus.

For the fault case where the second load gets a bus fault or an MPU fault then the base register is never restored and there is no dependence on which bus the instructions are being executed from.

When the base register is the second register in the LDRD list, of the form `LDRD Rb, Ra, [Ra, #imm]`, then this erratum cannot occur.

You will not be affected by this erratum if:

1) you do not execute code from the system bus and if your code bus does not generate bus faults and you do not execute LDRD's that cross MPU boundaries, or

2) if your compiler does not generate LDRD's

Conditions

Either:

1) An LDRD is being executed where the base register is in the list and write-back is not used:
`LDRD Ra, Rb, [Ra, #imm]`

2) Instructions and data are both being fetched via the system bus. This occurs for locations in memory greater than 0x20000000.

3) The first LDRD address is prioritised and issued to the system bus, whilst the instruction fetch is internally waited. The instruction fetch is issued to the system bus upon completion of the first part of the LDRD. The second part of the LDRD is issued to the system bus upon completion of the instruction fetch.

4) An interrupt occurs in between the two load operations.

Or:

1) An LDRD is being executed where the base register is in the list and write-back is not used:
`LDRD Ra, Rb, [Ra, #imm]`

2) A bus fault or MPU fault occurs for the second load.

Implications

The base register will not be restored as expected preventing the instruction from being restarted correctly upon return from the interrupt service routine or from the fault handler.

Workaround

There are two workarounds for this erratum. However, if the instructions are always executed from the code space and faults cannot occur then a workaround is not required.

The first workaround is to replace the LDRD instruction affected by this erratum with other suitable instructions.

```
 LDRD Ra, Rb, [Ra, #imm]
```

may be directly replaced by two LDR instructions which will produce exactly the same functionality:

```
 LDR Rb, [Ra, #imm + 4]
```

```
LDR Ra, [Ra, #imm]
```

Alternatively, an LDRD with base in list may still be used if the base register is the second register in the list:

```
LDRD Rb, Ra, [Ra, #imm]
```

However, in order to achieve the same functionality this requires that the data at the two addresses are swapped, or that the following instructions using Ra or Rb swap their source registers.

The second workaround can be applied when using Cortex-M3 r2p0. It is possible to prevent this erratum occurring for the interrupt case by setting DISMCYCINT (bit[0]) in the Auxiliary Control Register which is located at address 0xE000E008. This bit prevents the interruption of multi-cycle instructions and will therefore increase the interrupt latency of Cortex-M3.

Setting DISMCYCINT does not prevent the second load being faulted which means that the base will still be incorrect for bus faults or MPU faults.

Due to the performance impact of workaround two, and because workaround two does not address the fault conditions of this erratum, Arm recommends that workaround one is used.

641267: Bit-band access could read or write wrong bit in BE8

Category B

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0

Description

In some circumstances, if an access to a bit-band alias region is immediately followed by another access to a bit-band alias region which has a different access size compared to the first access then the wrong bit may be returned or modified when the core is used in big-endian mode.

Conditions

1) The core is running in big-endian mode (BIGEND input set at reset).
2) Bit-banding is utilised.
3) An access occurs to a bit-band alias region.
4) A second access occurs to a bit-band alias region immediately after the first one.
5) The two accesses have different size attributes.
6) The memory transaction inserts at least one wait state.

Implications

For store operations the wrong bit of the bit-band region may be modified. For load operations the wrong bit of the bit-band region may be read.

Workaround

A workaround is only required if big-endian mode is used, bit-banding is utilised and the memory has wait-states.

If this is the situation then two workarounds are possible:

1) Always access the bit-band alias with the same size attributes. Or
2) Stop consecutive bit-band alias accesses by inserting a NOP in between them.

740455: SVC and BusFault/MemManage may occur out of order

Category B

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0

Description

An SVC is a precise exception and it is generated by executing the SVC instruction. If the fetch for the instruction following an SVC instruction has been faulted, either externally or by the MPU, then the corresponding BusFault or MemManage fault handler should not be entered before the SVC handler as that instruction should not have been attempted to be executed. The SVC handler should be executed first and then if the next instruction is faulted when it is re-fetched it should enter the corresponding fault handler.

For Cortex-M3 the BusFault or MemManage handler may be entered before the SVC handler, if the priorites allow, otherwise the BusFault or MemManage handler will be tail-chained to. Neither of these events should occur and the instruction following the SVC should be re-fetched once the SVC handler has completed. Only if the instruction is faulted once more should the appropriate handler be entered.

Conditions

1) SVC is executed
2) The following instruction fetch is faulted, either externally or by the MPU

Implications

The MemManage or BusFault handler may be entered even though the faulted instruction which followed the SVC should not have been executed.

Workaround

A work around is only required if the SVC handler will not return to the return address that has been stacked for the SVC exception and the instruction access after the SVC will fault. If this is the case then padding can be inserted between the SVC and the faulting area of code, for example, by inserting NOP instructions.

752419: Interrupted loads to SP can cause erroneous behaviour

Category B

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0, r2p1

Description

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behaviour can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

1) LDR SP,[Rn],#imm
2) LDR SP,[Rn,#imm]!
3) LDR SP,[Rn,#imm]
4) LDR SP,[Rn]
5) LDR SP,[Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

1) LDR SP,[Rn],#imm
2) LDR SP,[Rn,#imm]!

Conditions

1) An LDR is executed, with SP/R13 as the destination
2) The address for the LDR is successfully issued to the memory system
3) An interrupt is taken before the data has been returned and written to the stack-pointer.

Implications

Unless the load is being performed to Device or Strongly-Ordered memory, there should be no implications from the repetition of the load. In the unlikely event that the load is being performed to Device or Strongly-Ordered memory, the repeated read can result in the final stack-pointer value being different than had only a single load been performed.

Interruption of the two write-back forms of the instruction can result in both the base register value and final stack-pointer value being incorrect. This can result in apparent stack corruption and subsequent unintended modification of memory.

Workaround

Both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

If repeated reads are acceptable, then the base-update issue may be worked around by performing the stack-pointer load without the base increment followed by a subsequent ADD or SUB instruction to perform the appropriate update to the base register.

789571: Flash patch control register value might be incorrect

Category B

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r2p0

Description

The FPB (FlashPatch and Breakpoint unit) implements hardware breakpoints and patches both code and data from Code space to System space. The number of comparators in the FPB is configurable:

- A full FPB unit has two literal comparators and six instruction comparators.
- A reduced FPB unit has two instruction comparators.
- If the FPB is not implemented at all, there are no comparators.

The FlashPatch Control register (FP_CTRL) at address 0xE0002000 includes fields NUM_CODE and NUM_LIT that identify the number of comparators implemented. Because of this erratum, when the reduced FPB configuration is implemented the fields NUM_CODE and NUM_LIT read as if the full configuration has been implemented.

Conditions

1) The reduced FPB unit is implemented (by configuring Cortex-M3 with a DEBUG_LVL of 1)
2) A debug tool reads register FP_CTRL to determine the number of comparators available.

Implications

The fields in register FP_CTRL that identify the number of comparators indicate, to a debug tool, that more comparators are available for flash-patching and breakpoints than are actually implemented.

Workaround

The true configuration of FPB implemented can be found from the value of register FP_COMP2 at address 0xE0002010 in the following way. First write any non-zero value to FP_COMP2, then read back the register value. The number of comparators available can be determined from the values of FP_COMP2 and FP_CTRL as follows:

- If the value of FP_COMP2 is non-zero then the full FPB is present, so all comparators have been implemented.
- If FP_COMP2 is zero then if FP_CTRL is non-zero the reduced set of two comparators has been implemented.
- If FP_COMP2 and FP_CTRL are both zero then the FPB has not been implemented and no comparators are present.

806421: ITM can deadlock when global timestamping enabled

Category B

Products Affected: Cortex-M3 with ETM.

Present in: r2p1

Description

The Cortex-M3 processor contains an optional Instrumentation Trace Macrocell (ITM). This can be used to generate trace data under software control, and is also used with the Data Watchpoint and Trace (DWT) module which generates event driven trace. From r2p1 of the processor, support for global timestamping was introduced. This allows count values from a system-wide counter to be included in the trace stream.

When connected directly to a CoreSight funnel (or other component which holds ATREADY low in the idle state), the ITM will stop presenting trace data to the ATB bus after generating a timestamp packet. In this condition, the ITM_TCR.BUSY register will indicate BUSY.

Once this condition occurs, a reset of the Cortex-M3 is necessary before new trace data can be generated by the ITM.

Timestamp packets which require a 5 byte GTS1 packet, or a GTS2 packet do not trigger this erratum. This generally only applies to the first timestamp which is generated.

Devices which use the Cortex-M optimized TPIU (CoreSight ID register values 0x923 and 0x9A1) are not affected by this erratum.

Conditions

- Cortex-M3 is configured with DWT present
- The ATB is connected directly to a CoreSight Funnel or similar component
- Global timestamping is enabled (ITM_TCR.GTSFREQ != b00)
- The ITM is enabled (ITM_TCR.ITMENA == 1)
- An event which causes a timestamp to be generated occurs.
- A 2nd event which causes a timestamp to be generated occurs with only bits [20:0] of the timestamp changing.

Implications

If the system is susceptible to this erratum, global timestamps cannot be used reliably.

Workaround

There is no software workaround for this erratum. If the device being used is susceptible to this erratum, you must not enable global timestamping.

A system implementer can add an ATB synchronous bridge slice component between the Cortex-M3 and the downstream ATB system.

2.6. Category B (Rare)

838469: Store immediate overlapping exception return operation might vector to incorrect interrupt

Category B Rare

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0, r2p1

Description

The Cortex-M3 includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in there being a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

Configurations affected

This erratum only affects systems where writeable memory locations can exhibit more than one wait state.

Conditions

1) The handler for interrupt A is being executed.
2) Interrupt B, of the same or lower priority than interrupt A, is pending.
3) A store with immediate offset instruction is executed to a bufferable location.
    - STR/STRH/STRB <Rt>, [<Rn>,#imm]
    - STR/STRH/STRB <Rt>, [<Rn>,#imm]!
    - STR/STRH/STRB <Rt>, [<Rn>],#imm
4) Any number of additional data-processing instructions can be executed.
5) A BX instruction is executed that causes an exception return.
6) The store data has wait states applied to it such that the data is accepted at least two cycles after the BX is executed.
    - Minimally this is two cycles if the store and the BX instruction have no additional instructions between them.
    - The number of wait states required to observe this erratum needs to be increased by the number of cycles between the store and the interrupt service routine exit instruction.
7) Before the bus accepts the buffered store data, another interrupt C is asserted which has the same or lower priority as A, but a greater priority than B.

Implications

The processor should execute interrupt handler C, and on completion of handler C should execute the handler for B. If the conditions above are met, then this erratum results in the processor erroneously clearing the pending state of interrupt C, and then executing the handler for B twice. The first time the handler for B is executed it will be at interrupt C's priority level. If interrupt C is pended by a level-based interrupt which is cleared by C's handler then interrupt C will be pended again after the handler for B has completed and the handler for C will be executed.

If interrupt C is level based, then this interrupt will eventually become re-pending and subsequently be handled. If interrupt C is a single pulse interrupt, then there is a possibility that this interrupt will be lost.

Workaround

In r2pX versions of Cortex-M3, for software not using the memory protection unit, this erratum can be worked around by setting DISDEFWBUF in the Auxiliary Control Register.

In all other cases, the erratum can be avoided by ensuring a DSB occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

ARMCC:

```
  ...
  __schedule_barrier();
  __asm{DSB};
  __schedule_barrier();
}
```

GCC:

```
  ...
  __asm volatile ("dsb 0xf":::"memory");
}
```

2.7. Category C

463763: BKPT in debug monitor mode can cause DFSR mismatch

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1

Description

A BKPT may be executed in debug monitor mode which will cause the debug monitor handler to be run but the Debug Fault Status Register (DFSR) at address 0xE000ED30 will not have bit 1 set to indicate the cause was a BKPT instruction. This will only occur if an interrupt other than the Debug Monitor is already being processed just before the BKPT is executed.

Conditions

1) C_DEBUGEN (bit 0) in the Debug Halting Control and Status Register at address 0xE000EDF0 is 0.
2) MON_EN (bit 16) in the Debug Exception and Monitor Control Register at address 0xE000EDFC is 1.
3) An enabled interrupt occurs two cycles before the BKPT is executed that causes a pre-emption.

Implications

The Debug Monitor handler may be entered without the DFSR revealing the cause of the handler being entered.

Workaround

Should a workaround be required, it can be deduced that if the DFSR does not have any bits set when the debug monitor has been entered then the cause must be due to this corner case and that it was the result of a BKPT.

463764: Core may freeze for SLEEPONEXIT single instruction ISR

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1

Description

The SLEEPONEXIT functionality causes the core to enter the sleep mode when the exit from the sole active interrupt occurs. This means that there are no more interrupts active and the exit would have caused a return to the thread.

It is possible for the core to become frozen if the SLEEPONEXIT functionality is used and the interrupt service routine (ISR) concerned only contains a single instruction. This freezing may occur if only one interrupt is active and it is pre-empted by an interrupt whose handler only contains the single instruction. This instruction must be a legal ISR exit instruction that takes one cycle to execute (either a BX or a BLX). In this case the unstacking would occur after the single instruction had been executed as normal to return to the now only active interrupt handler. However, once it has returned no more instructions will be processed and the core will be frozen. Any new pre-empting interrupt will unfreeze the processor.

Conditions

1) SLEEPONEXIT (bit 1) in the System Control Register at address 0xE000ED10 is set.
2) An interrupt occurs that causes a pre-emption of the current ISR which is the only interrupt that is currently active.
3) The interrupt service routine that is entered consists of only one instruction (either BX or BLX) which causes a legal exit from that ISR.

Implications

The core may freeze and stop processing instructions when it returns to the only currently active ISR. Note that a new interrupt that causes a pre-emption would cause the core to become unfrozen and behave correctly again.

Workaround

If the SLEEPONEXIT functionality is required then do not allow an ISR to contain only one instruction. If an empty ISR is used then insert a NOP before the exit instruction.

463769: Unaligned MPU fault during a write may cause the wrong data to be written to a successful first access

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1

Description

When an unaligned store is executed by Cortex-M3 the transaction is split up into either two or three aligned transactions forming constituent parts of the larger transaction. The MPU will check that these transactions are permitted and will block them if necessary. If an unaligned transaction occurs where it overlaps two MPU regions then each region relating to the part of the transaction that hits that region will be checked.

If an unaligned store occurs that crosses an MPU region boundary and has an MPU permission fault for the second region check but not for the first region then it is possible for the second component's data to be written for the first successful transaction in place of the first transaction's data. This can occur for writes to either the D-Code or system bus but will only occur if one or more wait-states are applied for the first component of the store.

Conditions

1) The full MPU is present and enabled with at least one region programmed and enabled.
2) An unaligned store is executed by the processor. The store can be to either the D-Code or the System bus.
3) The store crosses an MPU region boundary.
4) The first region lookup passes, the second region lookup fails.
5) One or more wait states are applied via HREADYS or HREADYD.

Implications

The wrong data will be stored to a permitted address. However, a MemManage fault will occur immediately pointing to the instruction that caused the fault. This may lead to the instruction being re-executed and the store occurring successfully if it is for non-device memory. This would mean that the previously stored data would be overwritten and the wrong value would never be seen. This may not be true for a shared memory system.

Workaround

A workaround is only required if the MPU is present and enabled. Either:

1) do not allow accesses to span more than one region or
2) do not allow unaligned accesses at all or
3) program the MPU correctly if applicable

511864: Cortex-M3 may fetch instructions using incorrect privilege on return from an exception

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1

Description

Whilst unstacking registers on return from an exception to a User-privilege thread, Cortex-M3 attempts to simultaneously prefetch the thread's instruction stream. Before the register unstacking is complete, upto the first three memory transactions used to perform instruction prefetching may be erroneously marked as Privileged. This may allow between three and six instructions from a Privileged-access-only region to be executed by a User-privilege thread.

Once fetched, the instructions are executed with User-privilege. Instruction fetches performed after register unstacking has completed will be performed with User-privilege. Both the register unstacking, and any data-transactions generated by executing the erroneously executed instructions will be performed and correctly marked as User-privilege.

Conditions

1) Exception return is executed

2) The exception return is to user code

Implications

User-privileged code may contrive a situation in order to allow execution of up to three words worth of instructions intended to be accessible to Privileged-only execution; however, execution of said instructions will always be performed with User-privilege, thus there are no additional capabilities provided to User-privilege through this erratum.

There exists a theoretical possibility that User-privilege code could use this erratum to allow limited extraction of code and or data from Privileged-access only memory.

Note that read sensitive Privileged-access only peripherals should always be placed in an XN region either via the default memory map, or via the optional memory-protection-unit. Alternatively such peripherals should ignore transactions with HPROT[0] indicating that the transaction is an instruction fetch.

Workaround

None.

532314: DWT CPI counter increments during sleep

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1

Description

The DWT contains a number of counters for the profiling of applications. The CPI counter is used to indicate the total number of clock cycles beyond the first cycle of each instruction. The counter is specified to not increment whilst the core is sleeping but for previous revisions it does increment. This results in sleep cycles being counted as program execution cycles.

Conditions

1) The CPI counter in the DWT is enabled
2) Core sleeps during profiling

Implications

Profiling information could be calculated incorrectly if the following calculation is used:

InstructionCount = CycleCount - (CPIcount+LSUcount+INTcount+SLEEPcount) + FOLDcount

Workaround

The number of sleep cycles given by SLEEPCNT can be subtracted from the CPI cycle count to obtain the correct CPI cycle information. Use the following equation:

InstructionCount = CycleCount - (CPIcount+LSUcount+INTcount) + FOLDcount

548721: Internal write buffer could be active whilst asleep

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1

Description

If a store immediate that is marked as not strongly ordered is used immediately before a WFE or WFI then the store may still be in progress when the core has asserted the SLEEPING signal. This will only occur if wait-states are applied to the store operation. This will not cause a problem unless the location that the store was accessing was using a free-running clock whilst a clock-gating cell has been used to gate FCLK to form HCLK.

Conditions

1) A store with immediate offset is executed
2) The store operation is allowed to be bufferable
3) The store is followed immediately by a WFI or WFE
4) Wait-states are used to delay the data-phase of the store
5) A clock-gate is used to produce HCLK which is gated when SLEEPING is asserted
6) The location the store was to is using a free-running version of the core clock (FCLK)

Implications

An imprecise error response could be missed if it is issued by the peripheral whilst the core is asleep when the peripheral is not using a gated clock but the core is. The stored data will be correct as the core will hold HWDATA at the correct value until it wakes from sleep and completes the transaction.

Workaround

A software workaround is to insert DSB instructions before any WFE and WFI instructions in the application code.

607266: ETM traces BKPT as an executed instruction

Category C

Products Affected: Cortex-M3 with ETM.

Present in: r1p1, r2p0

Description

When tracing program execution using the ETM an extra instruction is traced if an exception is caused by a software or hardware breakpoint or if the processor halts due to a software or hardware breakpoint.

Conditions

1) BKPT instruction is executed and causes the processor to halt or enter the debug monitor, in this case the BKPT instruction is incorrectly traced
2) A hardware breakpoint causes the processor to halt or enter the debug monitor, in this case the instruction which matches the hardware breakpoint is incorrectly traced
3) A hardware watchpoint matches and causes the processor to halt or enter the debug monitor, in this case the instruction which causes the hardware watchpoint to match is incorrectly traced

If the processor halts due to an External or Internal debug request or a Vector catch, the correct instructions are traced.

Implications

Trace analysis tools will incorrectly consider the extra instruction to have executed.

Workaround

This is a workaround for trace tool vendors.

If entry to the debug monitor is traced, the instruction traced immediately before the entry to the debug monitor was not executed and must be discarded.

If the processor halts due to a hardware or software breakpoint, the instruction traced immediately prior to the halt was not executed and must be discarded. The reason for halting can be determined from the Debug Fault Status Register (DFSR).

616516: Incorrect core feature identification registers

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0

Description

The following identification register fields return the wrong value when read:

1) Instruction Set Attributes Register0 (ID_ISAR0). Bits 19:16 should read as 0 instead of 4 as no coprocessor instructions are supported.
2) Instruction Set Attributes Register4 (ID_ISAR4). Bits 7:4 should read as 3 instead of 0 to indicate that constants shifts are supported on loads/stores and some DP operations.
3) Memory Model Feature register0 (ID_MMFR0). Bits 23:20 should read as 1 instead of 0 to indicate that an auxiliary control register is available.
4) Memory Model Feature register2 (ID_MMFR2). Bits 27:24 should read as 1 instead of 0 to indicate that wait for interrupt is supported.
5) Debug Features register0 (ID_DFR0). Bits 23:20 should read as 0 instead of 1 when no debug is present to indicate that the debug model is not supported.

Implications

The identification values that are read give the wrong information.

Workaround

No workaround is required. Identification fields are for information purposes only.

661722: External event might be lost when core is sleeping

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0

Description

An external event on the RXEV input pin may be lost if the core is sleeping following a `WFI` instruction and the core wakes up without entering an exception or debug mode. This can only occur during a WFI sleep if PRIMASK is set and FaultMask is clear and the asynchronous exception raised has a higher group priority than any active exception and a higher group priority than BASEPRI.

Conditions

1) The processor is sleeping following execution of a `WFI` instruction
2) An event is received on the RXEV input
3) An interrupt request is asserted after RXEV has been asserted
4) PRIMASK is set
5) FAULTMASK is clear
6) Priority of interrupt is the highest priority and would have activated if PRIMASK was 0
7) The processor executes a `WFE` instruction
8) No more events occur to cause a wake up event

Implications

The RXEV event is not stored and is missed. This means that when the core executes the next `WFE` instruction the core will sleep when it should stay awake.

Workaround

Do not use `WFE` and `WFI` instructions in combination or do not suppress interrupts using PRIMASK in the critical region.

If `WFI` and `WFE` are used in the same code sequence then insert a `SEV` instruction in between the `WFI` and the `WFE` instructions.

674118: TBH will never cause an alignment fault

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0

Description

For ARM v7M the following data accesses support unaligned addressing, and only generate alignment faults when the CCR.UNALIGN_TRP bit is set in the Configuration and Control Register:

1) Non halfword-aligned LDR{S}H{T} and STRH{T}
2) Non halfword-aligned TBH
3) Non word-aligned LDR{T} and STR{T}

Cortex-M3 does cause an alignment fault for the load and store cause but it does not generate a fault for the TBH case.

Conditions

1) A TBH instruction is executed
2) CCR.UNALIGN_TRP bit is set in the Configuration and Control Register at address 0xE000ED14
3) The address is not halfword aligned

Implications

An alignment fault will not be generated when it should have been. This will occur when the user is trying to ensure all accesses are aligned and sets the unaligned trap enable. With this errata the user may not spot that a TBH was unaligned.

Workaround

This erratum will only occur when the unaligned trap has been enabled and the user is trying to identify all unaligned accesses. Ensure by inspection that all TBH instructions are aligned as required.

Note: Clearing bit UNALIGN_TRP affects erratum 612616 "HPROT and MEMATTR incorrect on some unaligned transactions" as a proposed workaround consists in setting bit UNALIGN_TRP.

1320266: Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address

Category C

Products Affected: Cortex-M3, Cortex-M3 with ETM.

Present in: r1p1, r2p0, r2p1

Description

Normally, the debugger uses the Flash Patch and Breakpoint (FPB) unit for breakpoints or for patching ROM code during debugging. However, you can also use the FPB functionality as a method of in-the-field ROM code patching. On the Cortex-M3 processor, the processor reset can be asynchronously asserted and this could potentially cause problems if the processor is in the middle of writing to debug components (which are not reset by the processor reset) such as the FPB unit.

Configurations Affected

A Cortex-M3 processor that is configured with debug.

Conditions

1) The processor is programming the FPB to remap an address in ROM to fetch a replacement opcode or vector from an area in RAM.

2) The processor is asynchronously reset during the programming of the FPB.

3) The data is incorrectly written to the FPB register due to metastability.

Implications

In the unlikely event of this occurring it may cause incorrect functional operation of the processor to occur. If the FPB is programmed with incorrect data it may cause the processor to unintentionally patch instructions.

Workaround

The problem does not arise if the FPB is not used to patch instructions.

If the FPB is used to patch instructions, a software workaround is to ensure that the FPB is globally disabled before programming any comparators. If code exists which programs the FPB other than at reset, the software workaround should include:

1. Disable the FPB.
2. Program the individual comparators required.
3. Explicitly disable the individual comparators not required.
4. Re-enable the FPB.

This sequence prevents any corruptly programmed FPB comparators from being activated.