Arm Processor
Cortex™-M4 (AT520) and Cortex-M4 with FPU (AT521)

Product Revision r0

Software Developers Errata Notice

Non-Confidential - Released

Non-Confidential Proprietary notice

LES-PRE-20349

Document confidentiality status

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

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-0426805
- 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
12	New	839676	CatB	Rare	Fused MAC instructions give incorrect results for rare data combinations
13	New	1299509	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
10	New	806422	CatB		ITM can deadlock when global timestamping enabled
10	New	838869	CatB	Rare	Store immediate overlapping exception return operation might vector to incorrect interrupt

01 Apr 2013: Changes in Document v1

Page	Status	ID	Cat	Rare	Summary of Erratum
9	New	776924	CatB		VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Contents

Chapter 1.
Introduction

This chapter introduces the errata notice for the Arm processors Cortex-M4 and Cortex-M4 with Floating Point Unit.

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 **X** indicates that the erratum affects the revision shown at the top of that column.

This document includes errata that affect revision r0 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	r0p0	r0p1
806422	CatB		ITM can deadlock when global timestamping enabled	X	X
776924	CatB		VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	X	X
752770	CatB		Interrupted loads to SP can cause erroneous behaviour	X	X
838869	CatB	Rare	Store immediate overlapping exception return operation might vector to incorrect interrupt	X	X
839676	CatB	Rare	Fused MAC instructions give incorrect results for rare data combinations	X	X
1299509	CatC		Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address	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

752770: Interrupted loads to SP can cause erroneous behaviour

Category B

Products Affected: Cortex-M4, Cortex-M4F.

Present in: r0p0, r0p1

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.

776924: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

Category B

Products Affected: Cortex-M4F.

Present in: r0p0, r0p1

Description

On Cortex-M4 with FPU, the VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

Conditions

1) The floating point unit is present and enabled
2) Lazy context saving is not disabled
3) A VDIV or VSQRT is executed
4) The destination register for the VDIV or VSQRT is one of s0 - s15
5) An interrupt occurs and is taken
6) The interrupt service routine being executed does not contain a floating point instruction
7) 14 cycles after the VDIV or VSQRT is executed, an interrupt return is executed

A minimum of 12 of these 14 cycles are utilised for the context state stacking, which leaves 2 cycles for instructions inside the interrupt service routine, or 2 wait states applied to the entire stacking sequence (which means that it is not a constant wait state for every access).

In general this means that if the memory system inserts wait states for stack transactions then this erratum cannot be observed.

Implications

The VDIV or VQSRT instruction does not complete correctly and the register bank and FPSCR are not updated, meaning that these registers hold incorrect, out of date, data.

Workaround

A workaround is only required if the floating point unit is present and enabled. A workaround is not required if the memory system inserts one or more wait states to every stack transaction.

There are two workarounds:

1) Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).
2) Ensure that every interrupt service routine contains more than 2 instructions in addition to the exception return instruction.

806422: ITM can deadlock when global timestamping enabled

Category B

Products Affected: Cortex-M4, Cortex-M4F.

Present in: r0p0, r0p1

Description

The Cortex-M4 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. The processor supports global timestamping. 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-M4 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-M4 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 can not 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-M4 and the downstream ATB system.

2.6. Category B (Rare)

838869: Store immediate overlapping exception return operation might vector to incorrect interrupt

Category B Rare

Products Affected: Cortex-M4, Cortex-M4F.

Present in: r0p0, r0p1

Description

The Cortex-M4 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 once 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

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");
}
```

839676: Fused MAC instructions give incorrect results for rare data combinations

Category B Rare
Products Affected: Cortex-M4F.
Present in: r0p0, r0p1

Description

The Cortex-M4 processor includes optional floating-point logic which supports the fused MAC instructions (VFNMA, VFNMS, VFMA, VFMS). This erratum causes fused MAC operations on certain combinations of operands to result in either or both of the following:

- A result being generated which is one Unit of Least Precision (ULP) greater than the correct result.
- Inexact or underflow flags written to the Floating-point Status Control Register, FPSCR, incorrectly.

Configurations Affected

Cortex-M4F configured with floating-point.

Conditions

The conditions for this erratum are all of the following:

1. Cortex-M4 is configured with floating-point and it is enabled in the Coprocessor Access Control Register, CPACR.
2. Flush-to-Zero (FZ) mode is not enabled, that is, FPSCR.FZ is clear.
3. A fused MAC instruction (VFNMA, VFNMS, VFMA, or VFMS) is executed with all of the following properties:
    - The addition part of the operation is Unlike Signed Addition (USA). This implies that the combination of the instruction used and the signs of the operands means that a subtraction is being performed.
    - The result of the instruction, before rounding, is subnormal. This implies that the result is smaller in magnitude than 2^{-126}.
    - The significance of the product and the addend operand are the same or differ by one.

Implications

If this erratum occurs, then the processor either produces an incorrect result to a computation or fails to run a floating-point exception routine when it should.

Note:

It is expected that most algorithms only use normalised numbers because:

- Subnormal results have low precision.
- It is easier to avoid underflow.

This erratum does not affect algorithms which only use normalized numbers.

Workaround

Enable FZ mode in the FPSCR.

2.7. Category C

1299509: Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address

Category C

Products Affected: Cortex-M4, Cortex-M4F

Present in: r0p0, r0p1. Open.

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-M4 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-M4 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.