ARM1176JZ(F)-S™ Processors

r0p7 release

Software Developers Errata Notice

ARM®

ARM UAN 0014B (ID032713)

ARM1176JZ(F)-S Processors
Software Developers Errata Notice

Release Information

The following changes have been made to this book.

Change History

Date	Issue	Confidentiality	Change
15 October 2012	A	Non-confidential	First release, for r0p7 release
26 March 2013	B	Non-confidential	Second release, for r0p7 release

Proprietary Notice

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.

Product Status

The information in this document is final, that is for a developed product.

Web Address

http://www.arm.com

Contents
ARM1176JZ(F)-S Processors Software Developers Errata Notice

Chapter 1
Introduction

This chapter introduces the errata descriptions for ARM1176JZF-S™ and ARM1176JZ-S™ processors, for r0p7 releases.

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 1-1 Categorization of errata

Errata type	Definition
Category 1	Behavior that is impossible to work around and that severely restricts the use of the product in all applications, or the majority of applications, rendering the device unusable.
Category 2	Behavior that contravenes the specified behavior and that might limit or severely impair the intended use of specified features, but does not render the product unusable in all or the majority of applications.
Category 3	Behavior that was not the originally intended behavior but should not cause any problems in applications.

1.3 Errata summary

Table 1-2 lists all the errata described in this document. The Status column shows any errata that are new or updated in the current issue of the document. An erratum is shown as updated if there has been any change to the text of the erratum Description, Conditions, Implications or Workaround.

Table 1-2 List of errata

Status	ID	Area	Cat	Summary of erratum
-	714068	Prog	2	Prefetch Instruction Cache Line or Invalidate Instruction Cache Line by MVA can cause deadlock
-	716151	Prog	2	Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line
-	720013	Prog	2	Invalidate Instruction Cache operations can fail
-	760522	Prog	2	A BLX (immediate) instruction may corrupt instruction stream
-	328593	Prog	3	VFP can take an inexact exception on non-CDP VFP instructions
-	396798	Prog	3	Interrupted folded branch may be missed from ETM trace
-	434754	Prog	3	FAR/FSR write immediately following precise abort can corrupt FAR/FSR
Updated	486865	Prog	3	Wait For Interrupt can cause deadlock when either TCK or JTAGSYNCBYPASS are HIGH
-	727104	Prog	3	Prefetch Instruction Cache Line operation may corrupt instruction stream

Chapter 2
Errata Descriptions

This chapter includes the errata descriptions for ARM1176JZF-S™ and ARM1176JZ-S™ processors, for r0p7 releases.

2.1 Category 1

There are no errata in this category.

2.2 Category 2

This section describes the Category 2 errata.

2.2.1 (714068) Prefetch Instruction Cache Line or Invalidate Instruction Cache Line by MVA can cause deadlock

Status

Affects: product ARM1176JZ-S, ARM1176JZF-S.

Fault Type: Category 2

Description

The Prefetch Unit normally speculatively fetches instructions ahead of the current execution point in order to improve performance. The Prefetch Unit stops speculatively fetching instructions when it fetches an unconditional, UNPREDICTABLE branch. Speculatively fetching can be forced to continue after fetching such a branch by setting the PHD bit (bit [28]) of the Auxiliary Control Register.

The erratum occurs when a prefetch or invalidate by MVA operation is performed by writing to CP15 register 7. If an unconditional, unpredictable branch is fetched shortly afterwards then the PFU will stop speculatively fetching. Speculative fetching before the branch was detected might result in a page table walk. If the page table walk aborts then under rare circumstances the CP15 write instruction might enter a state where it is stalled until the abort is cleared. The abort cannot be cleared until the Prefetch Unit receives a new fetch address and resumes fetching. The core cannot send a new fetch address because the core is being stalled by the CP15 write instruction. Under these rare circumstances the processor will deadlock.

Conditions

The following conditions must all be met:

1. A Prefetch Instruction Cache Line (`MCR p15, 0, Rx, c7, c13, 1`) or Invalidate Instruction Cache Line by MVA (`MCR p15, 0, Rx, c7, c5, 1`) instruction is executed.
2. An unconditional, UNPREDICTABLE branch is located within the 32 bytes after the prefetch instruction.
3. The PHD bit of the Auxiliary Control Register is not set (the reset value).

In addition to the above conditions, this erratum requires specific timing between internal signals and on the external AXI buses. Therefore any code that replicates the given conditions might not stimulate this erratum.

Implications

The processor core might deadlock under the conditions described above.

Workaround

A workaround is to set the PHD bit of the Auxiliary Control Register to disable instruction prefetch halting on unconditional, unpredictable instructions:

```
MRC p15, 0, r0, c1, c0, 1   ; Read Auxiliary Control Register
ORR r0, r0, #0x10000000     ; Set PHD bit
MCR p15, 0, r0, c1, c0, 1   ; Write Auxiliary Control Register
MOV r0, #0
MCR p15, 0, r0, c7, c5, 4   ; Flush Prefetch Buffer
```

2.2.2 (716151) Clean Data Cache Line by MVA can corrupt subsequent stores to the same cache line

Status

Affects: product ARM1176JZ-S, ARM1176JZF-S.

Fault Type: Category 2

Description

Clean by MVA operations to the data cache can be performed by writing to CP15 register 7. These operations perform a cache lookup using the supplied MVA. If a cache line is hit, any dirty data in the cache line is written back to external memory and the cache line is marked as clean. If only half of the cache line is marked as dirty then only the data in that half will be written back.

Hit-Under-Miss allows program execution to continue after there has been a data cache miss. Hit-Under-Miss is enabled by default after reset and can only be disabled by setting the FI bit (bit [21]) in the CP15 Control Register.

The Hit-Under-Miss functionality makes it possible for a store instruction located after a Clean Data Cache Line by MVA instruction to write to the cache before the Clean completes. If the cache line is marked as half-dirty before the instructions are executed then the Clean will have to write the data in the dirty half of the cache line back to external memory. If both instructions access the same cache line and the store causes the entire cache line to be marked as dirty, it is possible for the Clean to still only write back half of the cache line to external memory. The data from the store will then be marked as clean in the cache line even though it may not be synchronized with the corresponding location in external memory, possibly resulting in data corruption if the cache line is later evicted or invalidated

Conditions

The following conditions must all be met:

1. Hit-Under-Miss is enabled.
2. A cache line exists that is valid and partially dirty.
3. A Clean Data Cache Line by MVA (`MCR p15, 0, Rx, c7, c10, 1`) instruction is executed to clean the cache line identified in condition 2.
4. A store instruction to the part of the cache line that was not dirty is executed at some point after the Clean instruction.

In addition to the above conditions, this erratum requires specific timing between internal signals and on the external AXI buses. Therefore any code that replicates the given conditions might not stimulate this erratum

Implications

The store data might be lost under the conditions described above, resulting in data corruption.

Clean and Invalidate Data Cache Line by MVA operations are not affected by this erratum.

Data cache evictions are not affected by this erratum.

Clean Data Cache Line by MVA operations are commonly followed by a DSB to ensure visibility of the cleaned data. This prevents the erratum from occurring.

Workaround

Two workarounds are possible for this erratum:

1. Execute a DMB (`MCR p15, 0, Rx, c7, c10, 5`) or DSB (`MCR p15, 0, Rx, c7, c10, 4`) between the Clean Data Cache Line by MVA instruction and the next store instruction to the cache line that was cleaned.

2. Disable the Hit-Under-Miss functionality by setting the FI bit in the CP15 Control Register. To avoid putting the processor into full low interrupt latency mode, the FIO bit (bit [31]) of the Auxiliary Control Register must also be set. This combination of control bits disables the Hit-Under-Miss functionality but has no effect on the interrupt latency.

The first workaround will result in a minor decrease in the performance of Clean instructions. The second workaround will decrease the instruction throughput when a data cache miss occurs, meaning that it will have a negligible effect on programs with effective data cache usage.

2.2.3 (720013) Invalidate Instruction Cache operations can fail

Status

Affects: product ARM1176JZ-S, ARM1176JZF-S

Fault Type: Category 2

Description

Under very rare conditions, Invalidate Instruction Cache entry operations can fail to invalidate a cache.

The following CP15 operations are affected:

1. Invalidate Instruction Cache line by MVA (`MCR p15, 0, <rd>, c7, c5, 1`)
2. Invalidate Instruction Cache range (`MCRR p15, 0, <end_addr>, <start_addr>, c5`)
3. Invalidate Instruction Cache line by set/way (`MCR p15, 0, <rd>, c7, c5, 2`)

Conditions

1. The cache line to be invalidated has been requested from the L2 memory but not committed to the instruction cache
2. An Invalidate Instruction Cache line or Invalidate Instruction cache range operation is executed

It should be noted that particular timings of events are required in addition to the above conditions for the erratum to be triggered. As a result, not all sequences that meet these conditions will trigger the erratum

Implications

Associative instruction cache invalidate operations might not correctly invalidate the instruction cache, which can lead to incorrect program execution.

The cache maintenance operations are expected to be contained within HAL code and require the processor to be in a privilege mode to be executed. This makes analyzing code and applying workarounds more straightforward.

Workaround

The following workarounds can be applied for this erratum:

1. For associative operations (Invalidate Instruction Cache line by MVA and Invalidate Instruction Cache range) execute the invalidate operation twice, following each invalidate operation with an Instruction Synchronisation Barrier (ISB) operation. For example:

   ```
   MCR p15, 0, R0, c7, c5, 1
   ```

   becomes:

   ```
   MCR p15, 0, R0, c7, c5, 1
   ISB
   MCR p15, 0, R0, c7, c5, 1
   ISB
   ```

   This workaround is not applicable to Invalidate Instruction Cache line by set/way operations.

2. Use the Invalidate Entire Instruction Cache instruction (`MCR p15, 0, <Rd>, c7, c5, 0`) instead. This workaround can be used to replace all three affected operations.

2.2.4 (760522) A BLX (immediate) instruction may corrupt instruction stream

Status

Affects: product ARM1176JZ-S, ARM1176JZF-S.

Fault Type: Category 2

Description

The ARM1176JZ(F)-S processor cores contain a prefetch unit that can fetch many instructions ahead of the current execution point in the instruction stream. It is capable of fetching up to two instructions each cycle from the Instruction Cache RAMs. In ARM state this requires a 64-bit doubleword fetch in order to get two 32-bit instructions. In Thumb state each instruction only occupies 16 bits, so the prefetch unit reduces power consumption by only fetching 32-bit words. The prefetch unit also contains a Branch Target Address Cache (BTAC) that is used to cache the target address of a branch and to predict whether a branch will be taken or not. This allows the prefetch unit to continue fetching instructions after it has fetched a predictable branch.

Under very rare conditions, a linefill to the target of a dynamically predicted `BLX` (immediate) instruction may cause the processor to incorrectly apply the Thumb state power optimization when fetching the branch target. This may cause the processor to fail to execute the ARM instruction located at the address of the `BLX` target + 4 and may execute a different instruction instead.

Conditions

The following conditions must all be met:

1. Dynamic branch prediction must be enabled.
2. The Instruction Cache must be enabled and the relevant instructions must be marked as cacheable in the page table.
3. A `BLX` (immediate) instruction must be executed in Thumb state to switch to ARM state.
4. The `BLX` (immediate) must either:
   a. be the target of a branch instruction, or
   b. be located at an address whose value modulo 32 is equal to 30 or 0 (`0x1E` or `0x00`).
5. The target of the `BLX` (immediate) must be a doubleword aligned address.
6. The target of the `BLX` (immediate) must not be an unconditional branch or trap instruction.

In addition to the above conditions, this erratum requires specific timing between internal signals and also specific timing on the external AXI buses. It also depends on the instructions that are in the Instruction Cache and the Branch Target Address Cache (BTAC). Therefore any code that replicates the given conditions might not stimulate this erratum.

Implications

The instruction located at the address of the `BLX` target + 4 may fail to execute and a different instruction may execute instead, resulting in UNPREDICTABLE behavior.

This erratum has been graded as category 2 as the evidence from several years of deployment of this core in production systems indicate that the occurrence of this erratum is sufficiently low that the erratum does not severely restrict the use of the product in all, or the majority of applications.

Workaround

One of the following software workarounds can be applied for this erratum:

1. Ensure that all Thumb `BLX` (immediate) instructions have targets that are not doubleword aligned.

2. Do not use the Thumb `BLX` (immediate) instruction. An equivalent sequence of instructions must be used instead, for example a `LDR <Rn>, =target_address` followed by a `BLX <Rn>`.

3. Disable dynamic branch prediction by clearing the DB bit (bit 1) of the Auxiliary Control Register. This may have a significant negative impact on the overall performance of the device.

2.3 Category 3

This section describes the Category 3 errata.

2.3.1 (328593) VFP can take an inexact exception on non-CDP VFP instructions

Status

Affects: product ARM1176JZF-S.

Fault Type: Category 3

Description

If the VFP FPSCR register IXE bit is set, all VFP `CDP` instructions are bounced to software using an Undefined instruction trap. This also sets the VFP EXC register EX bit to cause most other following VFP operations to be exceptional. Because of this erratum, if a VFP `CDP` instruction appears in the shadow of a branch or exception, and the IXE bit is set, then the EX bit is set erroneously. This causes subsequent VFP instructions (other than `CDP` operations) to be redirected incorrectly.

Conditions

1. The VFP is enabled
2. The IXE bit is set to enable inexact exceptions to be trapped
3. A VFP `CDP` operation (or equivalent bit pattern) appears in a branch or exception shadow

Implications

The use of the VFP IXE bit to trap all inexact exceptions results in all VFP arithmetic operations being handled by software support code. Therefore applications are unlikely to set the IXE bit.

Where the IXE bit is set, this results in some VFP load and store operations being redirected to the support code incorrectly. However, suitable modification of the support code can detect these cases, and so mask this erratum from all users

Workaround

In the unusual event of a workaround being needed, the support code can be modified so that presentation of a VFP instruction other than a `CDP`, when the IXE bit is set, results in the operation being re-run

2.3.2 (396798) Interrupted folded branch may be missed from ETM trace

Status

Affects: product ARM1176JZ-S, ARM1176JZF-S.

Fault Type: Category 3

Description

If an interrupt request occurs during an operation that cannot be interrupted, then the interrupt will normally be taken on the next instruction. However, if the next instruction is a folded branch, then in some circumstances the interrupt may be taken on the instruction following the folded instruction. This can occur when the branch prediction is known to be correct, and avoids having to re-execute the folded instruction when the interrupt handler returns. This gives an improvement in performance.

If this erratum occurs, the folded instruction will not be present in the ETM trace.

Conditions

1. Bit 21 of the CP15 control register is set to enable the low interrupt latency configuration.
2. An `MCR`, `MRC`, `LDC` or `STC` instruction to CP14 or CP15 is executed.
3. This is followed by a branch instruction.
4. The branch is folded onto the next instruction.
5. The branch is correctly predicted.
6. An IRQ or FIQ request occurs whilst the CP14/15 operation is in progress.
7. The ETM is in use to trace the instruction stream.

Implications

This erratum does not affect the behavior of the processor, it only affects what is reported to the ETM.

If the erratum occurs, the branch instruction will be missing from the ETM trace. If any ETM address comparators are programmed for the address of the branch instruction, they will not match. Because the missing instruction is caused by an interrupt, it will always be followed by an exception packet in the trace stream. Therefore, trace decompressors will not lose instruction synchronisation

Workaround

There is no need to work around this erratum.

2.3.3 (434754) FAR/FSR write immediately following precise abort can corrupt FAR/FSR

Status

Affects: product ARM1176JZ-S, ARM1176JZF-S.

Fault Type: Category 3

Description

A load to normal non-cacheable memory that generates an external abort updates the Data Fault Status Register and the Data Fault Address Register.

If the aborting load is followed in program order by a write to the Data Fault Status Register then the Data Fault Status Register may be corrupted.

If the aborting load is followed in program order by a write to the Data Fault Address Register then the Data Fault Address Register may be corrupted

Conditions

1. Bit 21 of the CP15 Control Register is set.
2. A load to Normal non-cacheable memory generates an external abort.
3. The load is immediately followed in program order by a write to the Data Fault Status Register or the Data Fault Address Register.

Implications

The code sequence for stimulating this erratum is unlikely to be encountered in applications. If the write to the FSR/FAR is resetting the register to `0x0` then this erratum will not be observed.

If encountered this erratum may result in the FAR containing an address that is not associated with the external abort and the FSR containing an incorrect or invalid value.

Workaround

To prevent this erratum occurring it must be ensured that all outstanding memory accesses should have completed before writing to the FSR/FAR. This can be accomplished by inserting a data memory barrier before writing to the FAR/FSR. For example:

```
MCR p15, 0, <Rd>, c7, c10, 5 ; Data Memory Barrier
MCR p15, 0, <Rd>, c5, c0, 0  ; FSR write
```

2.3.4 (486865) Wait For Interrupt can cause deadlock when either TCK or JTAGSYNCBYPASS are HIGH

Status

Affects: product ARM1176JZ-S, ARM1176JZF-S.

Fault Type: Category 3

Description

The ARM1176JZ-S and ARM1176JZF-S processor cores have a JTAG interface which includes a debug clock input, **TCK**. A debugger must drive this clock input in order to access the processor's debug features. This input is normally held low when no debugger is connected. The processor synchronizes the JTAG interface signals to the core clock domain internally. A debugger that externally synchronizes the JTAG interface signals to the core clock domain can hold **JTAGSYNCBYPASS** high to disable the internal synchronizers. The processor cores also have an **INTSYNCEN** input which can be tied high when synchronous interrupts are used. This reduces interrupt latency as the interrupt synchronizers are bypassed.

The Wait For Interrupt (WFI) operation (`MCR p15,0,<Rd>,c7,c0,4`) puts the processor into a low-power state and stops it executing more instructions until an interrupt (or debug) request occurs, regardless of whether the interrupts are disabled. This low-power state causes the core clock signal (CLKIN) to be gated from most of the processor core logic. When **TCK** or **JTAGSYNCBYPASS** is high the core clock signal is not gated because the processor core might have to perform an operation requested by the debugger. If **TCK** or **JTAGSYNCBYPASS** is high when an interrupt occurs after a WFI operation has been executed, the timing of the state machine which restarts the processor core is affected. This makes it possible for the processor core to deadlock when the interrupt occurs.

Conditions

The following conditions must all be met:

1. The **INTSYNCEN** signal must be high, which will disable the interrupt synchronization registers. The problem might also occur in some ARM1176 implementations that have **INTSYNCEN** low, because of timing problems with the internal interrupt synchronizers.
2. The **TCK** or **JTAGSYNCBYPASS** signal must be high. If the JTAG interface is being driven by an active debugger, the **TCK** signal will periodically go high.
3. A WFI operation must be executed.
4. An interrupt must occur.

In addition to the above conditions, this erratum requires specific timing between internal signals and depends on the state of the instruction cache. Therefore any code that replicates the given conditions might not stimulate this erratum. The necessary condition is that there must be a pending instruction fetch which has been signaled by the instruction cache to the AXI interface at the time that WFI executes, but which has not yet started on the ARM1176 AXI interface

Implications

The processor core might deadlock if it is being accessed by a debugger while executing code which uses WFI. The processor core might also deadlock in a system which ties **TCK** or **JTAGSYNCBYPASS** high while executing code which uses WFI.

Workaround

The system workaround for this erratum is to tie **INTSYNCEN** low to enable the interrupt synchronization registers. It is also necessary for the ARM1176 silicon implementer to externally synchronize interrupts and to use static timing analysis to verify that the core synchronizer cannot register the interrupt on an earlier clock cycle than the clock gate synchronizer register.

A system that does not require the JTAG interface can instead implement the second workaround which is to tie **DBGnTRST**, **TCK** and **JTAGSYNCBYPASS** low. The problem can never occur when **TCK** is low and this is the preferred workaround for production silicon.

In cases where a software workaround is required, it is possible to avoid the erratum conditions by replacing the use of the `WFI` instruction with the following sequence:

```
    .align 5
    mov   r0, #2
1:  subs  r0, #1
    nop
    mcreq  p15, 0, r0, c7, c0, 4   @ wait for interrupt
    nop
    nop
    nop
    bne 1b
```

This code must run from a cacheable location

2.3.5 (727104) Prefetch Instruction Cache Line operation may corrupt instruction stream

Status

Affects: product ARM1176JZ-S, ARM1176JZF-S.

Fault Type: Category 3

Description

Prefetch Instruction Cache Line operations are used to load the cache line at the specified MVA into the instruction cache. This is normally done as part of a sequence to lock a critical code sequence into the cache.

The erratum occurs when the prefetch address is the same as the address of a currently filling cache line. Depending on the state of the instruction cache it is possible for stale data from the cache to be combined with data from the linefill. This will corrupt the instruction stream seen by the processor and can lead to corruption of the program flow. The cache contents will not be corrupted as a result of this erratum.

Conditions

The following conditions must all be met:

1. An instruction line fill is underway
2. A Prefetch Instruction Cache Line instruction (`MCR p15, 0, Rx, c7, c13, 1`) is executed before the instruction linefill is completed.
3. The Prefetch Instruction Cache Line target MVA (`Rx`) is the same as the current instruction linefill

In addition to the above conditions, this erratum requires specific timing between internal signals and on the external AXI buses. Therefore any code that replicates the given conditions might not stimulate this erratum.

Implications

Program execution flow may be corrupted as a result of this erratum.

It should be noted that Prefetch Instruction Cache Line operations are typically used as part of a sequence to lock instructions into the instruction cache. This requires that the instructions used in the lockdown sequence are either in a non-cacheable region of memory, or are already locked down in the instruction cache.

Workaround

A workaround to this erratum is to ensure that Prefetch Instruction Cache Line operations are executed from non-cacheable memory. The prefetch target must not be reachable within 8 instructions of the Prefetch Instruction Cache Line operation.