Chip Errata for the i.MX 6SoloLite

This document details the silicon errata known at the time of publication for the i.MX 6SoloLite multimedia applications processors.

Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Changes
Rev. 4	06/2014	• Added ERR007805 • Added ERR007927
Rev. 3	11/2013	• Added the following: ERR007007, ERR007008, ERR007265, ERR007266 • Updated the following: ERR003778, ERR005313
Rev. 2.1	5/2013	Updated workaround of ERR006282 to say “None” (removed erroneously displayed table).
Rev. 2	5/2013	• Added the following errata: – ERR006282 – ERR006308
Rev. 1.1	2/2013	Restored pages omitted in Rev. 1.
Rev. 1	1/2013	• Added the following: – ERR006223 – ERR006259 – ERR006281 – ERR006287
Rev. 0	10/2012	Initial public release.

Figure 1 provides a cross-reference to match the revision code to the revision level marked on the device.

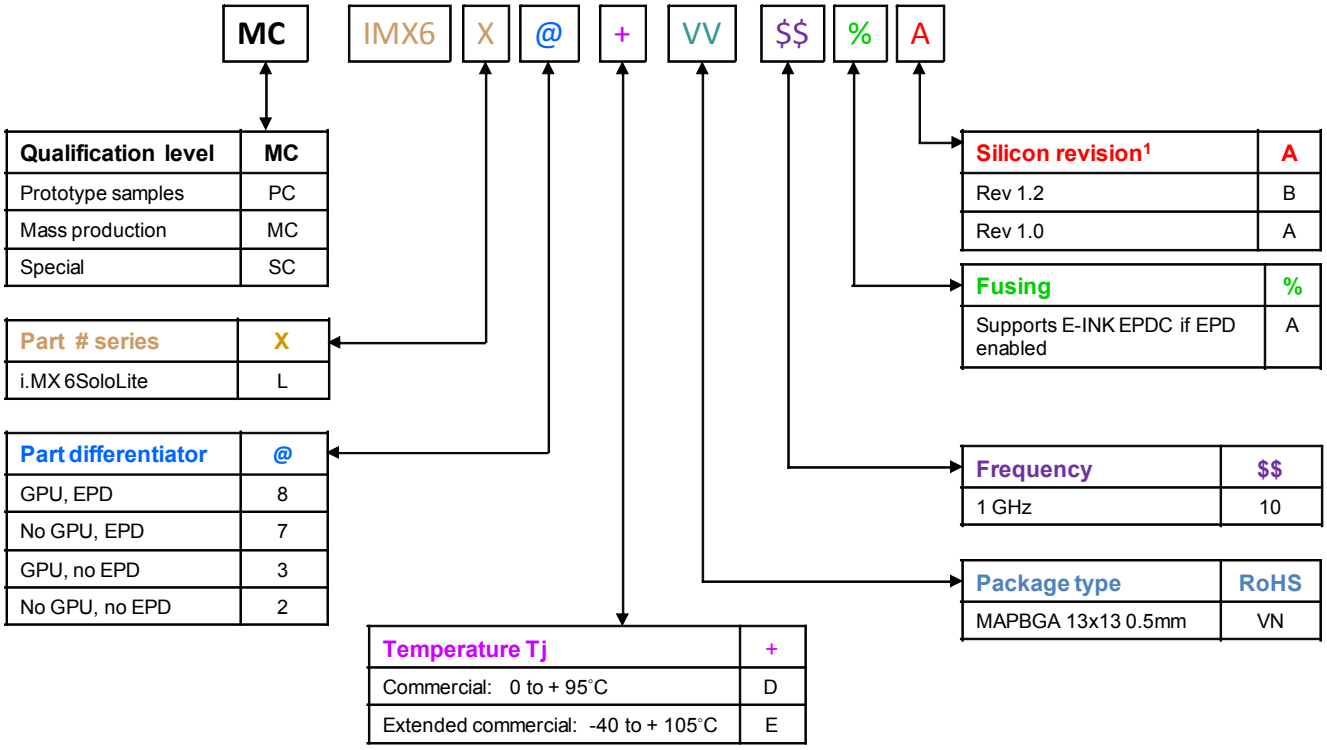

Figure 1. Revision Level to Part Marking Cross-Reference

For details on the ARM® configuration used on this chip (including ARM module revisions), please see the "Platform configuration" section of the "ARM Cortex®-A9 MPCore Platform" chapter of the *i.MX 6SoloLite Applications Processor Reference Manual*.

Table 2 summarizes errata on the i.MX 6SoloLite.

Table 2. Summary of Silicon Errata

Errata	Name	Solution	Page
Analog			
ERR005852	Analog: Transition from Deep Sleep Mode to LDO Bypass Mode may cause the slow response of the VDDARM_CAP output	No fix scheduled	7
ARM®			
ERR003717	ARM: 740657—Global Timer can send two interrupts for the same event	No fix scheduled	8
ERR003718	ARM: 743622—Faulty logic in the Store Buffer may lead to data corruption	No fix scheduled	10
ERR003719	ARM: 751469—Overflow in PMU counters may not be detected	No fix scheduled	12
ERR003721	ARM: 751473—Under very rare circumstances, Automatic Data prefetcher can lead to deadlock or data corruption	No fix scheduled	13
ERR003723	ARM: 751476—May miss a watchpoint on the second part of an unaligned access that crosses a page boundary	No fix scheduled	14
ERR003724	ARM: 754322—Possible faulty MMU translations following an ASID switch	No fix scheduled	15
ERR003725	ARM: 725631—ISB is counted in Performance Monitor events 0x0C and 0x0D	No fix scheduled	17
ERR003726	ARM: 729817—MainID register alias addresses are not mapped on Debug APB interface	No fix scheduled	18
ERR003727	ARM: 729818—In debug state, next instruction is stalled when sdabort flag is set, instead of being discarded	No fix scheduled	19
ERR003728	ARM: 740661—Event 0x74 / PMUEVENT[38:37] may be inaccurate	No fix scheduled	20
ERR003729	ARM: 740663—Event 0x68 / PMUEVENT[9:8] may be inaccurate	No fix scheduled	21
ERR003730	ARM: 743623—Bad interaction between a minimum of seven PLDs and one Non-Cacheable LDM can lead to a deadlock	No fix scheduled	23
ERR003731	ARM: 743626—An imprecise external abort, received while the processor enters WFI, may cause a processor deadlock	No fix scheduled	25
ERR003732	ARM: 751471—DBGPCSR format is incorrect	No fix scheduled	26
ERR003733	ARM: 751480—Conditional failed LDREXcc can set the exclusive monitor	No fix scheduled	28
ERR003734	ARM: 752519—An imprecise abort may be reported twice on non-cacheable reads	No fix scheduled	29
ERR003735	ARM: 754323—Repeated Store in the same cache line may delay the visibility of the Store	No fix scheduled	30
ERR003736	ARM: 756421—Sticky Pipeline Advance bit cannot be cleared from debug APB accesses	No fix scheduled	32
ERR003737	ARM: 757119—Some “Unallocated memory hint” instructions generate an UNDEFINED exception instead of being treated as NOP	No fix scheduled	33
ERR003738	ARM: 751475—Parity error may not be reported on full cache line access (eviction / coherent data transfer / cp15 clean operations)	No fix scheduled	34
ERR003739	ARM: 751470—Imprecise abort on the last data of a cache linefill may not be detected	No fix scheduled	35

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ERR003741	ARM/PL310: 729815—The “High Priority for SO and Dev reads” feature can cause Quality of Service issues to cacheable read transactions	No fix scheduled	36
ERR003743	ARM/PL310: 754670—A continuous write flow can stall a read targeting the same memory area	No fix scheduled	37
ERR004326	ARM/MP: 761321—MRC and MCR are not counted in event 0x68	No fix scheduled	38
ERR004327	ARM/MP: 764319—Read accesses to DBGPRSR and DBGOSLSR may generate an unexpected UNDEF	No fix scheduled	39
ERR005175	ARM/MP: 771221—PLD instructions may allocate data in the Data Cache regardless of the Cache Enable bit value	No fix scheduled	40
ERR005183	ARM/MP: 771224—Visibility of Debug Enable access rights to enable/disable tracing is not ensured by an ISB	No fix scheduled	41
ERR005185	ARM/MP: 771225—Speculative cacheable reads to aborting memory region clear the internal exclusive monitor, may lead to livelock	No fix scheduled	42
ERR005187	ARM/MP: 771223—Parity errors on BTAC and GHB are reported on PARITYFAIL[7:6], regardless of the Parity Enable bit value	No fix scheduled	44
ERR005198	ARM/PL310: 780370—DATAERR, TAGERR, and Tag parity errors are incorrectly sampled by the eviction buffer, leading to data corruption	No fix scheduled	45
ERR005382	ARM/MP: 775419—PMU event 0x0A (exception return) might count twice the LDM PC ^ instructions with base address register write-back	No fix scheduled	48
ERR005383	ARM/MP: 775420—A data cache maintenance operation that aborts, followed by an ISB and without any DSB in-between, might lead to deadlock	No fix scheduled	49
ERR005385	ARM/MP: 782772—A write to Strongly Ordered memory region, followed by a condition-failed LDREX, might deadlock the processor	No fix scheduled	50
ERR005386	ARM/MP: 782773—Updating a translation entry to move a page mapping might erroneously cause an unexpected translation fault	No fix scheduled	51
ERR005387	ARM/MP: 782774—A spurious event 0x63, “STREX passed,” can be reported on an LDREX that is preceded by a write to Strongly Ordered memory region	No fix scheduled	53
ERR005391	ARM: Debug CTI interrupt can cause a system deadlock when power gating the core	No fix scheduled	54
ERR006259	ARM: Debug/trace functions (PMU, PTM and ETB) are disabled with absence of JTAG_TCK clock after POR	No fix scheduled	55
ERR007007	ARM/MP: 794073 -- Speculative instruction fetches with MMU disabled might not comply with architectural requirements	No fix scheduled	56
ERR007008	ARM/MP: 794074 --A write request to Uncacheable Shareable memory region might be executed twice	No fix scheduled	57
CCM			
ERR005311	CCM: After exit from WAIT mode, unwanted interrupt(s) taken during WAIT mode entry process could cause cache memory corruption	No fix scheduled	59
ERR006223	CCM: Failure to resume from Wait/Stop mode with power gating	No fix scheduled	60

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ERR007265	CCM: When improper low-power sequence is used, the SoC enters low power mode before the ARM core executes WFI	No fix scheduled	61
eLCDIF			
ERR006287	eLCDIF/EPDC/PXP/SPDC: Display domain modules register read may return value for the prior read access after resuming from power gating	No fix scheduled	62
EPDC			
ERR004573	EPDC: Collision status must be read before clearing IRQ	No fix scheduled	63
ERR005313	EPDC: Incorrect data fetched when the buffer update width is 2048 pixels or greater	No fix scheduled	64
EXSC			
ERR004365	EXSC: Exclusive accesses to certain memories are not supported to full AXI specification	No fix scheduled	65
ERR005828	EXSC: Protecting the EIM memory map region causes unpredictable behavior	No fix scheduled	66
GPU			
ERR005908	GPU2D: Image quality degradation observed for stretch blits when the stretch factor is exactly an integer	No fix scheduled	67
I2C			
ERR007805	I2C: When the I2C clock speed is configured for 400 kHz, the SCL low period violates the I2C specification	No Fix scheduled	68
I/O			
ERR004307	I/O: USB_HSIC interface should not be configured to Differential input mode	No fix scheduled	68
MMDC			
ERR005778	MMDC: DDR Controller's measure unit may return an incorrect value when operating below 100 MHz	No fix scheduled	70
ROM			
ERR007927	ROM: 32 kHz internal oscillator timing inaccuracy may affect SD/MMC and OneNAND boot	No fix scheduled	72
ERR005645	ROM: Normal SD clock speed (SDR12) not selectable in SD/SDXC boot mode	No fix scheduled	74
ERR005768	ROM: In rare cases, secondary image boot flow may not work due to mis-sampling of the WDOG reset	No fix scheduled	75
ERR006282	ROM code uses nonreset PFDs to generate clocks, which may lead to random boot failures	Fixed in silicon revision 1.2.	80
ERR007266	ROM: EIM NOR boot may fail if plug-in is used	No fix scheduled	81
SSI			
ERR003778	SSI: In AC97, 16-bit mode, received data is shifted by 4-bit locations	No fix scheduled	76

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ERR006281	USB: Incorrect DP/DN state when only VBUS is applied	No fix scheduled	77
ERR006308	USB: Host non-doubleword –aligned buffer address can cause host to hang on OUT Retry		78
uSDHC			
ERR004536	uSDHC: ADMA Length Mismatch Error occurs if the AHB read access is slow, when reading data from the card	No fix scheduled	79

ERR005852 Analog: Transition from Deep Sleep Mode to LDO Bypass Mode may cause the slow response of the VDDARM_CAP output

Description:

Normally, the VDDARM_CAP supply takes only approximately 40 μs to raise to the correct voltage when exiting from Deep Sleep (DSM) mode, if the LDO is enabled. If the LDO bypass mode is selected, the VDDARM_CAP supply voltage will drop to approximately 0 V when entering and when exiting from DSM, even though the VDDARM_IN supply is already stable, the VDDARM_CAP supply will take about 2 ms to rise to the correct voltage.

Projected Impact:

ARM core might fail to resume.

Workarounds:

The software workaround to prevent this issue it to switch to analog bypass mode (0x1E), prior to entering DSM, and then, revert to the normal bypass mode, when exiting from DSM.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER3

ERR003717 ARM: 740657—Global Timer can send two interrupts for the same event

Description:

The Global Timer can be programmed to generate an interrupt request to the processor when it reaches a given programmed value. Due to the erratum, when the Global Timer is programmed not to use the auto-increment feature, it might generate two interrupt requests instead of one.

Conditions:

The Global Timer Control register is programmed with the following settings:

- Bit[3] = 1'b0 – Global Timer is programmed in "single-shot" mode
- Bit[2] = 1'b1 – Global Timer IRQ generation is enabled
- Bit[1] = 1'b1 – Global Timer value comparison with Comparator registers is enabled
- Bit[0] = 1'b1 – Global Timer count is enabled

With these settings, an IRQ is generated to the processor when the Global Timer value reaches the value programmed in the Comparator registers.

The Interrupt Handler then performs the following sequence:

1. Read the ICCIAR (Interrupt Acknowledge) register
2. Clear the Global Timer flag
3. Modify the comparator value to set it to a higher value
4. Write the ICCEOIR (End of Interrupt) register

Under these conditions, due to the erratum, the Global Timer might generate a second (spurious) interrupt request to the processor at the end of this Interrupt Handler sequence.

Projected Impact:

The erratum creates spurious interrupt requests in the system.

Workarounds:

Because the erratum only happens when the Global Timer is programmed in "single-shot" mode, that is, when it does not use the auto-increment feature, a first possible workaround could be to program the Global Timer to use the auto-increment feature.

If this solution does not work, a second workaround could be to modify the Interrupt Handler to avoid the offending sequence. This is achieved by clearing the Global Timer flag after having incremented the Comparator register value.

Then, the correct code sequence for the Interrupt Handler should look as below:

1. Read the ICCIAR (Interrupt Acknowledge) register
2. Modify the comparator value to set it to a higher value
3. Clear the Global Timer flag
4. Clear the Pending Status information for Interrupt 27 (Global Timer interrupt) in the Distributor of the Interrupt Controller.

5. Write the ICCEOIR (End of Interrupt) register

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP. The BSP does not use ARM global timer.

ERR003718 ARM: 743622—Faulty logic in the Store Buffer may lead to data corruption

Description:

Under very rare conditions, a faulty optimization in the Cortex®-A9 store buffer might lead to data corruption.

Conditions:

The code sequence which exhibits the failure requires at least five cacheable writes in 64-bit data chunk:

- Three of the writes must be in the same cache line
- Another write must be in a different cache line
- All of the above four writes hit in the L1 data cache
- A fifth write is required in any of the above two cache lines that fully writes a 64-bit data chunk

With the above code sequence, under very rare circumstances, this fifth write might get corrupted, with the written data either being lost, or being written in another cache line.

The conditions under which the erratum can occur are extremely rare, and require the coincidence of multiple events and states in the Cortex-A9 micro-architecture.

As an example: let's assume A, A', A'', and A''' are all in the same cache line—B and B' are in another cache line. The following code sequence might trigger the erratum:

```
STR A
STR A′
STR A′′
STR B
STR A′′′ (or STR B′)
```

At the time where the first four STR are in the Cortex-A9 store buffer, and the fifth STR arrives at a very precise cycle in the Store Buffer input stage, then the fifth STR might not see its cache line dependency on the previous STR instructions. Because of this, in cases when the cache line A or B gets invalidated due to a coherent request from another CPU, the fifth STR might write in a faulty cache line, causing data corruption.

An alternative version of the erratum might happen even without a coherent request — In the case when the fifth STR is a 64-bit write in the same location as one of A, A', A'', then the erratum might also be exhibited. Note that this is a quite uncommon scenario because it requires a first write to a memory location that is immediately and fully overwritten.

Projected Impact:

When it occurs, this erratum creates a data corruption.

Workarounds:

A software workaround is available for this erratum that requires setting bit[6] in the undocumented Diagnostic Control register, placed in CP15 c15 0 c0 1.

The bit can be written in Secure state only, with the following Read/Modify/Write code sequence:

```
MRC p15,0,rt,c15,c0,1
ORR rt,rt,#0x40
MCR p15,0,rt,c15,c0,1
```

When this bit is set, the “fast lookup” optimization in the Store Buffer is disabled, which will prevent the failure to happen.

Setting this bit has no visible impact on the overall performance or power consumption of the processor.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER3

ERR003719 ARM: 751469—Overflow in PMU counters may not be detected

Description:

Overflow detection logic in the Performance Monitor Counters is faulty, and under certain timing conditions, the overflow might remain undetected. In this case, the Overflow Flag Status Register (PMOVSR) is not updated as it should, and no interrupt is reported on the corresponding PMUIRQ line.

Projected Impact:

PMU overflow detection is not reliable.

Workarounds:

The workaround for this erratum involves setting two PMU counters to count the same event, and explicitly offset them by 1 at the start of the count. This can be achieved with the following sequence:

1. Disable PMU count
2. Set Counter0 to value N where N is an arbitrary count value
3. Set Counter1 to value (N+1)
4. Enable PMU count

This ensures that at least one of the two counters will detect the overflow. Most of the time, each of the two counters will trigger, so using this workaround requires the software to reset both counters when the first one triggers.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround to be applied in a future BSP release

ERR003721 ARM: 751473—Under very rare circumstances, Automatic Data prefetcher can lead to deadlock or data corruption

Description:

Under very rare timing circumstances, the automatic Data prefetcher might cause address hazard issues, possibly leading to a data corruption or a deadlock of the processor.

Conditions:

The erratum can only happen when the Data Cache and MMU are enabled in the following cases:

- On all memory regions marked as Write-Back Non-Shared, when the Data Prefetcher in L1 is enabled (ACTLR[2]=1’b1), regardless of the ACTLR.SMP bit.
- On all memory regions marked as Write-Back Shared, when the Data Prefetch Hint in L2 is enabled (ACTLR[1]=1’b1), and when the processor is in SMP mode (ACTLR.SMP=1’b1).

Projected Impact:

When the bug happens, a data corruption or a processor deadlock can happen.

Workarounds:

The workaround for this erratum requires not enabling the automatic Data Prefetcher by keeping ACTRL[2:1]=2’b00, which is the default value on exit from reset.

Although this feature might show significant performance gain on a few synthetic benchmarks, it usually has no impact on real systems. It means, this workaround is not expected to cause any visible impact on final products.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER3. Linux BSP keeps ACTRL[2:1]=2’b00.

ERR003723 ARM: 751476—May miss a watchpoint on the second part of an unaligned access that crosses a page boundary

Description:

Under rare conditions, a watchpoint on the second part of an unaligned access that crosses a 4 KB page boundary and that is missed in the micro-TLB for the second part of its request might be undetected.

The erratum requires a previous conditional instruction that accesses the second 4 KB memory region (= where the watchpoint is set), is missed in the micro-TLB, and is condition failed. The erratum also requires that no other micro-TLB miss occurs between this conditional failed instruction and the unaligned access. This implies that the unaligned access must hit in the micro-TLB for the first part of its request.

Projected Impact:

A valid watchpoint trigger is missed.

Workarounds:

In case, a watchpoint is set on any of the first 3 bytes of a 4 KB memory region, and unaligned accesses are not being faulted, then the erratum might happen.

The workaround then requires setting a guard watchpoint on the last byte of the previous page, and dealing with any “false positive” matches as and when they occur.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003724 ARM: 754322—Possible faulty MMU translations following an ASID switch

Description:

A micro-TLB entry might be corrupted following an ASID switch, possibly corrupting subsequent MMU translations. The issue requires that a speculative explicit memory access is executed, but is speculative failed. The speculation fails if the memory access occurred under a mispredicted branch or in case it is conditional and condition failed.

This speculative memory access might miss in the TLB, and cause a Page Table Walk. The erratum occurs when the Page Table Walk starts, prior to the ASID switch code sequence, but completes afterwards.

In this case, the microTLB will get a new entry allocated with this new TLB entry, corresponding to the “old” ASID. The issue is that the micro-TLB does not register the ASID value, so that MMU translations which should happen with the new ASID following the ASID switch might hit in this stale micro-TLB entry, and get corrupted.

It is, however, important to note that there is no Trustzone Security risks because the Security state of the access is registered in the micro-TLB, and consequently, cannot be corrupted.

Projected Impact:

The errata might cause MMU translation corruptions.

Workarounds:

The workaround for this erratum involves adding a DSB in the ASID switch code sequence. The ARM architecture only mandates ISB before and after the ASID switch. Adding a DSB prior to the ASID switch ensures that the Page Table Walk completes prior to the ASID change, so that no stale entry can be allocated in the micro-TLB.

The examples in the ARM Architecture Reference Manual for synchronizing the change in the ASID and TTBR need to be changed as follows:

The sequence:

```
Change ASID to 0
ISB
Change Translation Table Base Register
ISB
Change ASID to new value
```

becomes

```
DSB
Change ASID to 0
ISB
Change Translation Table Base Register
ISB
DSB
Change ASID to new value
```

the sequence:

```
Change Translation Table Base Register to the global-only mappings
```

```
ISB
Change ASID to new value
ISB
Change Translation Table Base Register to new value
```

becomes

```
Change Translation Table Base Register to the global-only mappings
ISB
DSB
Change ASID to new value
ISB
Change Translation Table Base Register to new value
```

and the sequence:

```
Set TTBCR.PD0 = 1
ISB
Change ASID to new value
Change Translation Table Base Register to new value
ISB
Set TTBCR.PD0 = 0
```

becomes

```
Set TTBCR.PD0 = 1
ISB
DSB
Change ASID to new value
Change Translation Table Base Register to new value
ISB
Set TTBCR.PD0 = 0
```

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER3

ERR003725 ARM: 725631—ISB is counted in Performance Monitor events 0x0C and 0x0D

Description:

The ISB is implemented as a branch in the Cortex-A9 micro-architecture. This implies that events 0x0C (software change of PC) and 0x0D (immediate branch) are asserted when an ISB occurs. This is not compliant with the ARM architecture.

Projected Impact:

The count of events 0x0C and 0x0D are not 100% precise when using the Performance Monitor counters, due to the ISB being counted in addition to the real software changes to PC (for 0x0C) and immediate branches (0x0D).

The erratum also causes the corresponding PMUEVENT bits to toggle in case an ISB is executed.

- PMUEVENT[13] relates to event 0x0C
- PMUEVENT[14] relates to event 0x0D

Workarounds:

Count ISB instructions along with event 0x90. The user should subtract this ISB count from the results obtained in events 0x0C and 0x0D, to obtain the precise count of software change of PC (0x0C) and immediate branches (0x0D).

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003726 ARM: 729817—MainID register alias addresses are not mapped on Debug APB interface

Description:

The ARM Debug Architecture specifies registers 838 and 839 as “Alias of the MainID register”. They should be accessible through the APB Debug interface at addresses 0xD18 and 0xD1C.

In Cortex-A9, the two alias addresses are not implemented. A read access at any of these two addresses returns 0, instead of the MIDR value.

Note that read accesses to these two registers through the internal CP14 interface are trapped to UNDEF, which is compliant with the ARM Debug architecture. So, the erratum only applies to the alias addresses through the external Debug APB interface.

Projected Impact:

If the debugger or any other external agent tries to read the MIDR register using the alias addresses, it will get a faulty answer (0x0), which can cause all sorts of malfunction in the debugger afterwards.

Workarounds:

The workaround for this erratum requires always accessing the MIDR at its original address, 0xD00, and not at any of its alias addresses.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR003727 ARM: 729818—In debug state, next instruction is stalled when sdabort flag is set, instead of being discarded

Description:

When the processor is in debug state, an instruction written to the ITR after a Load/Store instruction that aborts gets executed on clearing the SDABORT_l, instead of being discarded.

Projected Impact:

Different failures can happen due to the instruction being executed when it should not. In most cases, it is expected that the failure will not cause any significant problem.

Workarounds:

There are a selection of workarounds with increasing complexity and decreasing impact. In each case, the impact is a loss of performance when debugging:

- Do not use stall mode
- Do not use stall mode when doing load/store operations
- Always check for a sticky abort after issuing a load/store operation in stall mode (the cost of this probably means the above second workaround is a preferred alternative)
- Always check for a sticky abort after issuing a load/store operation in stall mode, before issuing any further instructions that might corrupt important target state (such as, further load/store instructions, instructions that write to “live” registers [VFP, CP15, etc.])

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR003728 ARM: 740661—Event 0x74 / PMUEVENT[38:37] may be inaccurate

Description:

Event 0x74 counts the total number of Neon instructions passing through the register rename pipeline stage. Due to the erratum, the “stall” information is not taken into account. So, one Neon instruction that remains for n cycles in the register rename stage is counted as n Neon instructions. As a consequence, the count of event 0x74 might be corrupted, and cannot be relied upon. The event is also reported externally on PMUEVENT[38:37], which suffers from the same inaccuracy.

Projected Impact:

The implication of this erratum is that Neon instructions cannot be counted reliably in the versions of the product that are affected by this erratum.

Workarounds:

No workaround is possible to achieve the required functionality of counting how many Neon instructions are executed (or renamed) in the processor.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003729 ARM: 740663—Event 0x68 / PMUEVENT[9:8] may be inaccurate

Description:

Event 0x68 counts the total number of instructions passing through the register rename pipeline stage. Under certain conditions, some branch-related instructions might pass through this pipeline stage without being counted. As a consequence, event 0x68 might be inaccurate, lower than expected. The event is also reported externally on PMUEVENT[9:8], which suffers from the same inaccuracy.

Conditions:

The erratum occurs when the following conditions are met:

- Events are enabled
- One of the PMU counters is programmed to count event 0x68 — number of instructions passing through the register rename stage. Alternatively, an external component counts, or relies on, PMUEVENT[9:8].
- A program, containing the following instructions, is executed:
  - A Branch immediate, without Link
  - An ISB instruction
  - An HB instruction, without Link and without parameter, in Thumb2EE state
  - An ENTERX or LEAVEX instruction, in Thumb2 or Thumb2EE state
- The program executed is causing some stalls in the processor pipeline

Under certain timing conditions specific to the Cortex-A9 micro-architecture, a cycle stall in the processor pipeline might “hide” the instructions mentioned above, thus ending with a corrupted count for event 0x68, or a corrupted value on PMUEVENT[9:8] during this given cycle. If the “hidden” instruction appears in a loop, the count difference can be significant.

As an example, let’s consider the following loop:

```
loop mcr 15, 0, r2, cr9, cr12, {4}
    adds r3, #1
    cmp.w r3, #loop_number
    bne.n loop
```

The loop contains four instructions; so, the final instruction count should (approximately) be four times the number of executed loops. In practice, the MCR is causing a pipeline stall that “hides” the branch instruction (bne.n); so, only three instructions are counted per loop, and the final count appears as three times the number of executed loops.

Projected Impact:

The implication of this erratum is that the values of event 0x68 and PMUEVENT[9:8] are imprecise, and cannot be relied upon.

Workarounds:

No workaround is possible to achieve the required functionality of counting how many instructions are precisely passing through the register rename pipeline stage.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003730 ARM: 743623—Bad interaction between a minimum of seven PLDs and one Non-Cacheable LDM can lead to a deadlock

Description:

Under very rare circumstances, a deadlock can happen in the processor when it is handling a minimum of seven PLD instructions, shortly followed by one LDM to an uncacheable memory location.

The LDM is treated as uncacheable in the following cases:

- The LDM is performed while the Data Cache is OFF
- The LDM is targeting a memory region marked as Strongly Ordered, Device, Normal Memory Non-Cacheable, or Normal Memory Write-Through
- The LDM is targeting a memory region marked as Shareable Normal Memory Write-Back, and the CPU is in AMP mode.

Conditions:

The code sequence that exhibits this erratum requires at least seven PLDs, shortly followed by one LDM, to an uncacheable memory region. The erratum happens when the LDM appears on the AXI bus before any of the seven PLDs. This can possibly happen if the first PLD is a miss in the micro-TLB; in that case, it needs to perform a TLB request which might not be serviced immediately because the mainTLB is already performing a Page Table Walk for another resource (for example, instruction side), or because the PLD request itself to the mainTLB is missing and causing a Page Table Walk.

Also note that the above conditions are not sufficient to recreate the failure, as additional rare conditions on the internal state of the processor are necessary to exhibit the errata.

Projected Impact:

The erratum might create a processor deadlock. However, the conditions that are required for this to occur are extremely unlikely to occur in real code sequences.

Workarounds:

The primary workaround might be to avoid the offending code sequence, that is, not to use uncacheable LDM when making intensive use of PLD instructions.

In case the above workaround cannot be done, another workaround for this erratum can be to set bit[20] in the undocumented Control register, which is placed in CP15 c15 0 c0 1.

This bit needs to be written with the following Read/Modify/Write code sequence:

```
MRC p15,0,r0,c15,c0,1
ORR r0,r0,#0x00100000
MCR p15,0,r0,c15,c0,1
```

Setting this bit causes all PLD instructions to be treated as NOPs, with the consequence that code sequences usually using the PLDs, such as the memcpy() routine, might suffer from a visible performance drop.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR003731 ARM: 743626—An imprecise external abort, received while the processor enters WFI, may cause a processor deadlock

Description:

An imprecise external abort received while the processor is ready to enter into WFI state might cause a processor deadlock.

Explicit memory transactions can be completed by inserting a DSB before the WFI instruction. However, this does not prevent memory accesses generated by previously issued PLD instructions page table walks associated with previously issued PLD instructions or as a result of the PLE engine.

If an external abort is returned as a result of one of these memory accesses after executing a WFI instruction, the processor can cause a deadlock.

Projected Impact:

In case, the non-explicit memory request receives an external imprecise abort response while the processor is ready to enter into WFI state, the processor might cause a deadlock.

In practical systems, it is not expected that these memory transactions will generate an external abort, as external aborts are usually a sign of significant corruption in the system.

Workarounds:

A possible workaround for this erratum is to protect all memory regions that can return an imprecise external abort with the correct MMU settings, to prevent any external aborts.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003732 ARM: 751471—DBGPCSR format is incorrect

Description:

About the DBGPCSR register, the ARM architecture specifies that:

- DBGPCSR[31:2] contains sampled value of bits [31:2] of the PC.
  The sampled value is an instruction address plus an offset that depends on the processor instruction set state.
- DBGPCSR[1:0] contains the meaning of PC sample value, with the following permitted values:
  - 0b00 ((DBGPCSR[31:2] << 2) - 8) references an ARM state instruction
  - 0bx1 ((DBGPCSR[31:1] << 1) - 4) references a Thumb or ThumbEE state instruction
  - 0b10 IMPLEMENTATION DEFINED

This field encodes the processor instruction set state, so that the profiling tool can calculate the true instruction address by subtracting the appropriate offset from the value sampled in bits [31:2] of the register.

In Cortex-A9, the DBGPCSR samples the target address of executed branches (but possibly still speculative to data aborts), with the following encodings:

- DBGPCSR[31:2] contains the address of the target branch instruction, with no offset
- DBGPCSR[1:0] contains the execution state of the target branch instruction:
  - 0xb00 for an ARM state instruction
  - 0xb01 for a Thumb2 state instruction
  - 0xb10 for a Jazelle state instruction
  - 0xb11 for a Thumb2EE state instruction

Projected Impact:

The implication of this erratum is that the debugger tools neither rely on the architected description for the value of DBGPCSR[1:0], nor remove any offset from DBGPCSR[31:2], to obtain the expected PC value.

Subtracting 4 or 8 to the DBGPCSR[31:2] value would lead to an area of code that is unlikely to have been recently executed, or that could even not contain any executable code.

The same might be true for Thumb instructions at half-word boundaries, in which case PC[1]=1 but DBGPCSR[1]=0, or ThumbEE instructions at word boundaries, with PC[1]=0 and DBGPCSR[1]=1.

In Cortex-A9, because the DBGPCSR is always a branch target (= start of a basic block to the tool), the debugger should be able to spot many of these cases and attribute the sample to the right basic block.

Workarounds:

The debugger tools can find the expected PC value and instruction state by reading the DBGPCSR register, and consider it as described in the Description section.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR003733 ARM: 751480—Conditional failed LDREXcc can set the exclusive monitor

Description:

A conditional LDREX might set the internal exclusive monitor of the Cortex-A9 even when its condition fails. So, any subsequent STREX that depends on this LDREXcc might succeed when it should not.

Projected Impact:

The implication of the erratum is that a subsequent STREX might succeed when it should not. So, the memory region protected by the exclusive mechanism can be corrupted if another agent is accessing it at the same time.

Workarounds:

The workaround for this erratum can be not to use conditional LDREX along with non-conditional STREX.

- If no conditional LDREX is used, the erratum cannot be triggered.
- If conditional LDREX is used, the associated STREX should be conditional too with the same condition, so that even if the exclusive monitor is set by the condition failed LDREX, the following STREX will not be executed because it will be condition failed too. For most situations this will naturally be the case anyway.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP. The BSP does not use conditional LDREX.

ERR003734 ARM: 752519—An imprecise abort may be reported twice on non-cacheable reads

Description:

When two outstanding read memory requests to device or non-cacheable normal memory regions are issued by the Cortex-A9, and the first one receives an imprecise external abort, then the second access might falsely report an imprecise external abort.

Conditions:

The erratum can only happen in systems which can generate imprecise external aborts on device or non-cacheable normal memory regions accesses.

Projected Impact:

When the erratum occurs, a second, spurious imprecise abort might be reported to the core when it should not.

In practice, the failure is not expected to cause any significant issues to the system because imprecise aborts are usually unrecoverable failures. Because the spurious abort can only happen following a first imprecise abort—either the first abort is ignored and the spurious abort is then ignored too, or it is acknowledged and probably generates a critical failure in the system.

Workarounds:

There is no practical software workaround for the failure.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003735 ARM: 754323—Repeated Store in the same cache line may delay the visibility of the Store

Description:

The Cortex-A9 implements a small counter that ensures the external visibility of all stores in a finite amount of time, causing an eventual drain of the merging store buffer. This is to avoid an earlier issue, where written data could potentially remain indefinitely in the Store Buffer.

This store buffer has merging capabilities, and will continue merging data as long as the write accesses are performed in the same cache line. The issue that causes this erratum is that the draining counter is reset each time a new data merging is performed.

When a code sequence is looping, and keeps on writing data in the same cache line, then the external visibility of the written data might not be ensured.

A livelock situation might consequently occur in case any external agent is relying on the visibility of the written data, and that the writing processor cannot be interrupted while doing its writing loop.

Conditions:

The erratum can only happen on normal memory regions.

Two example scenario, which might trigger the erratum, are described below:

- The processor keeps on incrementing a counter: writing the same word at the same address. The external agent (possibly another processor) is polling on this address, waiting for any update of the counter value to proceed.
  The store buffer will keep on merging the updated value of the counter in its cache line, so that the external agent will never see any updated value, possibly leading to livelock.
- The processor writes a value in a given word to indicate completion of its task, then keeps on writing data in an adjacent word in the same cache line.
  The external agent keeps on polling the first word memory location to check when the processor has completed its task. The situation is the same as above, as the cache line might remain indefinitely in the merging store buffer, creating a possible livelock in the system.

Projected Impact:

This erratum might create performance issues, or worst case livelock scenario, in case the external agent relies on the automatic visibility of the written data in a finite amount of time.

Workarounds:

The recommended workaround for this erratum involves inserting a DMB operation after the faulty write operation in code sequences that might be affected by this erratum. This ensures visibility of the written data by any external agent.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER3

ERR003736 ARM: 756421—Sticky Pipeline Advance bit cannot be cleared from debug APB accesses

Description:

The Sticky Pipeline Advance bit is bit[25] of the DBGDSCR register. This bit enables the debugger to detect whether the processor is idle. This bit is set to 1 every time the processor pipeline retires one instruction.

A write to DBGDRCR[3] clears this bit.

The erratum is that the Cortex-A9 does not implement any debug APB access to DBGDRCR[3].

Projected Impact:

Due to the erratum, the Sticky Pipeline Advance bit in the DBGDSCR cannot be cleared by the external debugger. In practice, this makes the Sticky Pipeline Advance bit concept unusable on Cortex-A9 processors.

Workarounds:

There is no practical workaround for this erratum. The only possible way to reset the Sticky Pipeline Advance bit is to assert the nDBGRESET input pin on the processor. This obviously has the side effect to reset all debug resources in the concerned processor, and any other additional Coresight components nDBGRESET is connected to.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003737 ARM: 757119—Some “Unallocated memory hint” instructions generate an UNDEFINED exception instead of being treated as NOP

Description:

The ARM architecture specifies that ARM opcodes of the form 11110 100x001 xxxx xxxx xxxx xxxx xxxx are “Unallocated memory hint (treat as NOP)” if the core supports the MP extensions, as the Cortex-A9 does.

The errata is that the Cortex-A9 generates an UNDEFINED exception when bits [15:12] of the instruction encoding are different from 4’b1111, instead of treating the instruction as a NOP.

Projected Impact:

Due to the erratum, an unexpected UNDEFINED exception might be generated. In practice, this erratum is not expected to cause any significant issue, as such instruction encodings are not supposed to be generated by any compiler, nor used by any handcrafted program.

Workarounds:

A possible workaround for this erratum is to modify the instruction encoding with bits[15:12]=4.b1111, so that the instruction is truly treated as a NOP by the Cortex-A9.

If the instruction encoding cannot be modified, the UNDEFINED exception handler has to cope with this case, and emulate the expected behavior of the instruction, that is, do nothing (NOP), before returning to normal program execution.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR003738 ARM: 751475—Parity error may not be reported on full cache line access (eviction / coherent data transfer / cp15 clean operations)

Description:

In the Data Cache, parity error detection is faulty. Parity error might not be detected when the line exits from the Data Cache, due to a line replacement, or due to a coherent request from another processor or from the ACP, or because of a CP15 cache clean operation.

Projected Impact:

Due to the erratum, a corrupted line might be evicted or transferred from the processor without the parity error being detected and reported.

Workarounds:

There is no workaround for this erratum.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003739 ARM: 751470—Imprecise abort on the last data of a cache linefill may not be detected

Description:

Data linefills are returned as 4-beat bursts of 64-bit data on the AXI bus. When the first three beat of data are valid, and the fourth one aborts, then the abort is not detected by the processor logic and no abort exception is taken. The processor then behaves as if no abort is reported on the line. It can allocate the line in its Data Cache, and use the aborted data during its program flow.

Conditions:

The processor needs to work with Data Cache enabled, and access some cacheable memory regions (Write Back, either Shared or Non-Shared).

The memory system underneath the processor needs to be able to generate aborts in this memory region, and must be able to generate aborts with a granularity smaller than the cache line.

Projected Impact:

When the erratum triggers, the processor does not detect the abort, so it might use some invalid (aborted) data without entering the Data Abort exception handler as it should normally do.

Workarounds:

None

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003741 ARM/PL310: 729815—The “High Priority for SO and Dev reads” feature can cause Quality of Service issues to cacheable read transactions

Description:

The “High Priority for SO and Dev reads” feature can be enabled by setting the bit[10] of the PL310 Auxiliary Control Register to 1. When enabled, it gives priority to Strongly Ordered and Device reads over cacheable reads in the PL310 AXI master interfaces. When PL310 receives a continuous flow of SO or Device reads, this can prevent cacheable reads, which are misses in the L2 cache, from being issued to the L3 memory system.

Conditions:

The erratum occurs when the following conditions are met:

- The bit[10] “High Priority for SO and Dev reads enable” of the PL310 Auxiliary Control Register is set to 1
- PL310 receives a cacheable read that is a miss in the L2 cache
- PL310 receives a continuous flow of SO or Device reads that take all address slots in the master interface

Projected Impact:

When the above conditions are met, the linefill resulting from the L2 cache miss is not issued till the flow of SO/Device reads stops. Note that each PL310 master interface has four address slots, so that the Quality of Service issue only appears on the cacheable read, if the L1 is able to issue at least four outstanding SO/Device reads.

Workarounds:

A workaround is only necessary in systems that are able to issue a continuous flow of SO or Device reads. In such a case, the workaround is to disable the “High Priority for SO and Dev reads” feature. This is the behavior by default.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR003743 ARM/PL310: 754670—A continuous write flow can stall a read targeting the same memory area

Description:

In the ARM L2 cache controller, PL310, hazard checking is done on bits [31:5] of the address.

When a read with Normal Memory (cacheable or not) attributes is received by PL310, hazard checking is performed with the active writes of the store buffer. If an address matching is detected, the read is stalled till the write completes.

Due to this erratum, a continuous flow of writes can stall a read targeting the same memory area.

Conditions:

The erratum occurs when the following conditions are met:

- PL310 receives a continuous write traffic targeting the same address marked with Normal Memory attributes
- While treating this flow, PL310 receives a read targeting the same 32-byte memory area

Projected Impact:

When the conditions above are met, the read might be stalled till the write flow stops.

Note that this erratum does not lead to any data corruption.

Note also that normal software code is not expected to contain long write sequence like the one causing this erratum to occur.

Workarounds:

There is no workaround for this erratum. A workaround is not expected to be necessary for this erratum either.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR004326 ARM/MP: 761321—MRC and MCR are not counted in event 0x68

Description:

Event 0x68 counts the total number of instructions passing through the Register rename pipeline stage. The erratum is that MRC and MCR instructions are not counted in this event.

The event is also reported externally on PMUEVENT[9:8], which suffers from the same defect.

Projected Impact:

The implication of this erratum is that the values of event 0x68 and PMUEVENT[9:8] are imprecise, omitting the number of MCR and MRC instructions. The inaccuracy of the total count depends on the rate of MRC and MCR instructions in the code.

Workarounds:

No workaround is possible to achieve the required functionality of counting how many instructions are precisely passing through the register rename pipeline stage, when the code contains some MRC or MCR instructions.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR004327 ARM/MP: 764319—Read accesses to DBGPRSR and DBGOSLSR may generate an unexpected UNDEF

Description:

CP14 read accesses to the DBGPRSR and DBGOSLSR registers generate an unexpected UNDEFINED exception when the DBGSWENABLE external pin is set to 0, even when the CP14 accesses are performed from a privileged mode.

Projected Impact:

Due to the erratum, the DBGPRSR and DBGOSLSR registers are not accessible when DBGSWENABLE=0.

This is, however, not expected to cause any significant issue in Cortex-A9 based systems because these accesses are mainly intended to be used as part of debug over power-down sequences, which is not a feature supported by the Cortex-A9.

Workarounds:

The workaround for this erratum consists in temporarily setting the DBGSWENABLE bit to 1 so that the DBGPRSR and DBGOSLSR registers can be accessed as expected.

There is no other workaround for this erratum.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005175 ARM/MP: 771221—PLD instructions may allocate data in the Data Cache regardless of the Cache Enable bit value

Description:

Preload Data (PLD) instructions prefetch and allocate any data marked as Write-Back (either Write-Allocate or Non-Write-Allocate, Shared or Non-Shared), regardless of the processor configuration settings, including the Data Cache Enable bit value.

Projected Impact:

Due to this erratum, unexpected memory cacheability aliasing is created which might result in various data consistency issues.

In practice, this erratum is not expected to cause any significant issue. The Data Cache is expected to be enabled as soon as possible in most systems, and not dynamically modified. So, only boot-up code would possibly be impacted by this erratum, but such code is usually carefully controlled and not expected to contain any PLD instruction while Data Cache is not enabled.

Workarounds:

In the case where a system is impacted by this erratum, a software workaround is available which consists in setting bit [20] in the undocumented Control register, which is placed in CP15 c15 0 c0 1.

This bit needs to be written with the following Read/Modify/Write code sequence:

MRC p15,0,r0,c15,c0,1

ORR r0,r0,#0x00100000

MCR p15,0,r0,c15,c0,1

Setting this bit causes all PLD instructions to be treated as NOPs, with the consequence that code sequences usually using the PLDs, such as the memcpy() routine, might suffer from a visible performance drop. So, if this workaround is applied, ARM strongly recommends restricting its usage to periods of time where the Data Cache is disabled.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005183 ARM/MP: 771224—Visibility of Debug Enable access rights to enable/disable tracing is not ensured by an ISB

Description:

According to the ARM architecture, any change in the Authentication Status Register should be made visible to the processor after an exception entry or return, or an ISB.

Although this is correctly achieved for all debug-related features, the ISB is not sufficient to make the changes visible to the trace flow. As a consequence, the WPTTRACEPROHIBITEDn signal(s) remain stuck to their old value up to the next exception entry or return, or to the next serial branch, even when an ISB is executed.

A serial branch is one of the following:

- Data processing to PC with the S bit set (for example, MOVS pc, r14)
- LDM pc ^

Projected Impact:

Due to the erratum, the trace flow might not start or stop, as expected by the program.

Workarounds:

To work around the erratum, the ISB must be replaced by one of the events causing the change to be visible. In particular, replacing the ISB by a MOVS PC to the next instruction will achieve the correct functionality.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005185 ARM/MP: 771225—Speculative cacheable reads to aborting memory region clear the internal exclusive monitor, may lead to livelock

Description:

On Cortex-A9, when a cacheable read receives an external abort, the aborted line is allocated as invalid in the Data Cache, and any allocation in the Data Cache clears the internal exclusive monitor.

So, if a program executes a LDREX/STREX loop which keeps on receiving an abort answer in the middle of the LDREX/STREX sequence, then the LDREX/STREX sequence never succeeds, leading to a possible processor livelock.

As an example, the following code sequence might exhibit the erratum:

loop LDREX

...

DSB

STREX

CMP

BNE loop

....

LDR (into aborting region)

The LDREX/STREX does not succeed on the first pass of the loop, and the BNE is mispredicted, so, the LDR afterwards is speculatively executed.

So, the processor keeps on executing:

LDR to aborting region (this speculative LDR now appears “before” the LDREX and DSB)

LDREX

DSB

STREX

The LDR misses in L1, and never gets allocated as valid because it is aborting

The LDREX is executed, and sets the exclusive monitor

The DSB is executed. It waits for the LDR to complete, which aborts, causing an allocation (as invalid) in the Data Cache, which clears the exclusive monitor

The STREX is executed, but the exclusive monitor is now cleared, so the STREX fails

The BNE might be mispredicted again, so the LDR is speculatively executed again, and the code loops back on the same failing LDREX/STREX sequence.

Conditions:

The erratum happens in systems which might generate external aborts in answer to cacheable memory requests.

Projected Impact:

If the program reaches a stable state where the internal exclusive monitor keeps on being cleared in the middle of the LDREX/STREX sequence, then the processor might encounter a livelock situation.

In practice, this scenario seems very unlikely to happen because several conditions might prevent the erratum from happening:

- Usual LDREX/STREX code sequences do not contain any DSB, so that it is very unlikely that the system would return the abort answer precisely in the middle of the LDREX/STREX sequence on each iteration.
- Some external irritators (for example, interrupts) might happen and cause timing changes which might exit the processor from its livelock situation.
- Branch prediction is very usually enabled, so the final branch in the loop will usually be correctly predicted after a few iterations of the loop, preventing the speculative LDR to be issued, so that the next iteration of the LDREX/STREX sequence will succeed.

Workarounds:

The following two workarounds are available for this erratum:

- Turn on the branch prediction.
- Remove the DSB in the middle of the LDREX/STREX sequence. If a DSB is truly required, it is strongly recommended to place it before the LDREX/STREX sequence, and implement the LDREX/STREX sequence as recommended by the ARM architecture.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005187 ARM/MP: 771223—Parity errors on BTAC and GHB are reported on PARITYFAIL[7:6], regardless of the Parity Enable bit value

Description:

PARITYFAIL signal bits [7] and [6] are expected to report parity errors occurring on the BTAC and GHB RAMs, when the parity error detection logic is enabled (ACTLR[9]=1’b1).

The erratum is that the Parity Enable bit, ACTLR[9], is not taken into account by the logic driving PARITYFAIL[7:6]. As a consequence, any parity error on the BTAC or GHB RAM will be reported on PARITYFAIL[7] or [6], even when parity error detection is not enabled.

Conditions:

The erratum happens on all configurations that have implemented parity support on the BTAC or GHB RAMs when dynamic branch prediction is enabled (SCTLR[11]=1’b1).

Projected Impact:

Due to the erratum, unexpected parity errors might be reported when parity is not enabled, if any parity error happens on the BTAC or GHB RAMs.

Note that implementing parity error detection is not mandatory on the BTAC and GHB RAMs because such errors might cause a branch mispredict, but no functional failure.

In systems which are implementing parity error detection on the BTAC and GHB RAMs, the erratum is not expected to cause any significant issue because parity is likely to be enabled very soon in the boot process.

Workarounds:

Because parity errors on the BTAC and GHB RAMs are not reported when the dynamic branch prediction is not enabled, the workaround consists in enabling parity error detection (ACTLR[9]), prior to enabling dynamic branch prediction (SCTLR[11]).

In systems where branch prediction is enabled while parity error detection remains disabled, the workaround consists in ignoring any assertion on the PARITYFAIL[7:6] bits.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005198 ARM/PL310: 780370—DATAERR, TAGERR, and Tag parity errors are incorrectly sampled by the eviction buffer, leading to data corruption

Description:

The PL310 L2 cache controller implements error logic to indicate errors have occurred when accessing the L2 cache RAM array. The following error information is available when accessing the RAM array:

- DATAERR (or DATAERR[3:0] if data banking is implemented) from Data RAM
- TAGERR[7:0] (or TAGERR[15:0] if 16 ways are implemented) from Tag RAM
- Parity error on Tag or Data RAM if parity is implemented

This information is associated with each individual RAM access, and is only meant to be sampled by the PL310 internal access requestor at precise cycles, depending on the programmable latencies of the accessed RAM (see Technical Reference Manual (TRM) for more information on RAM latencies).

More specifically, when an eviction is handled by the PL310 eviction buffer, both Tag and Data RAMs are accessed to get the whole eviction information. When either DATAERR or TAGERR is asserted high, or a tag parity error is detected during that process, the error information is captured by the eviction buffer, which cancels the corresponding eviction as a result.

Due to this erratum, the eviction buffer can incorrectly sample error information. As a result, an eviction can be wrongly cancelled and dirty data can be lost, leading to data corruption.

Note that data parity error is not part of this erratum. The reason is that this type of error information is not taken into account by the eviction buffer. This means that an eviction is always sent to the L3 memory system, regardless of whether a Data parity error has been detected or not, when accessing its data in the L2 cache.

Conditions:

The erratum occurs when the following conditions are met:

- The L2 cache contains dirty cache lines
- The eviction buffer accesses Tag and Data RAMs to get dirty cache line information before replacement
- While the eviction buffer accesses the RAMs, a tag parity error is detected, or DATAERR or TAGERR are asserted HIGH, but this error information is not meant to be captured by the eviction buffer (it may be directed to another PL310 block or DATAERR may be transiently asserted high before the end of the Data RAM latency period)
- The eviction buffer incorrectly samples the error information and cancels the corresponding eviction

Projected Impact:

When the above conditions are met, dirty data can be lost, leading to data corruption.

The implications of this data corruption depend on the error information and the PL310 configuration. All cases listed below need to be carefully assessed to know the exact impact of the erratum on a particular system.

DATAERR

In a system where DATAERR is tied low, this erratum does not apply as far as DATAERR is concerned.

In a system not implementing banking on the Data RAM and not driving DATAERR constantly low, the eviction buffer can sample transient and unstable high values of DATAERR, even if there is actually no expected error reported to PL310. This case is the most serious consequence of this erratum because it leads to a silent data corruption without any actual data error.

In a system using DATAERR for indicating Data RAM error and implementing banking on the Data RAM, the eviction buffer can only sample a true error coming from the Data RAM. However, this error may actually target another PL310 sub-block or another eviction slot. The erratum can thus still lead to data corruption, but the latter must be put in perspective relative to the true data error the overall system is facing. This is up to the system to assess how serious the data corruption is compared to the RAM error.

TAGERR

In a system where TAGERR is tied low, this erratum does not apply as far as TAGERR is concerned.

In a system using TAGERR for indicating Tag RAM error, the eviction buffer can only sample a true error coming from the Tag RAM. However, this error may actually target another PL310 sub-block or another eviction slot. The erratum can thus still lead to data corruption, but the latter must be put in perspective relative to the true tag error the overall system is facing. This is up to the system to assess how serious the data corruption is compared to the RAM error.

Tag parity error

In a system not implementing parity configuration in PL310, this erratum does not apply as far as the tag parity error is concerned.

In a system implementing parity, the eviction buffer can only sample a true tag parity error detected by the PL310 parity logic. However, this error may actually target another PL310 sub-block or another eviction slot. The erratum can thus still lead to a data corruption, but the latter must be put in perspective relative to the true parity error the overall system is facing. This is up to the system to assess how serious the data corruption is compared to the error.

Workarounds:

The following two software workarounds are available for systems affected by this erratum:

- Use write-through memory attributes for all cacheable accesses targeting PL310.
- Disable the logic responsible for generating RAM errors. This can imply disabling parity in PL310 and/or disabling DATAERR and TAGERR generation in the RAM array, depending on the implementation.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005382 ARM/MP: 775419—PMU event 0x0A (exception return) might count twice the LDM PC ^ instructions with base address register write-back

Description:

The LDM PC ^ instructions with base address register write-back might be counted twice in the PMU event 0x0A, which is counting the number of exception returns.

The associated PMUEVENT[11] signal is also affected by this erratum, and might be asserted twice by a single LDM PC ^ instruction with base address register write-back.

Projected Impact:

Due to the erratum, the count of exception returns is imprecise. The error rate depends on the ratio between exception returns of the form LDM PC ^ with base address register write-back and the total number of exceptions returns.

Workarounds:

There is no workaround to this erratum.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR005383 ARM/MP: 775420—A data cache maintenance operation that aborts, followed by an ISB and without any DSB in-between, might lead to deadlock

Description:

Under certain micro-architectural circumstances, a data cache maintenance operation that aborts, followed by an ISB and with no DSB occurring between these events, might lead to processor deadlock.

Conditions:

The erratum occurs when the following conditions are met:

- Some write operations are handled by the processor, and take a long time to complete. The typical situation is when the write operation (STR, STM, …) has missed in the L1 Data Cache.
- No memory barrier (DMB or DSB) is inserted between the write operation and the data cache maintenance operation mentioned in condition 3.
- A data cache maintenance operation is performed, which aborts due to its MMU settings.
- No memory barrier (DMB or DSB) is inserted between the data cache maintenance operation in previous condition and the ISB in next condition. Any other kind of code can be executed here, starting with the abort exception handler, following the aborted cache maintenance operation.
- An ISB instruction is executed by the processor.
- No memory barrier (DMB or DSB) is inserted between the ISB in previous condition and the read or write operation in next condition.
- A read or write operation is executed.

With the above conditions, an internal “Data Side drain request” signal might remain sticky, causing the ISB to wait for the Data Side to be empty, which never happens because the last read or write operation waits for the ISB to complete.

Projected Impact:

The erratum can lead to processor deadlock.

Workarounds:

A simple workaround for this erratum is to add a DSB at the beginning of the abort exception handler.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005385 ARM/MP: 782772—A write to Strongly Ordered memory region, followed by a condition-failed LDREX, might deadlock the processor

Description:

Under certain timing circumstances specific to the Cortex-A9 micro-architecture, a processor might deadlock when the execution of a write to a Strongly Ordered memory region is followed by the execution of a conditional, LDREX instruction that fails its condition code check.

Conditions:

The erratum occurs when the following conditions are met:

- The processor executes a write instruction to a Strongly Ordered memory region.
- The processor executes a conditional LDREX instruction.
- The instruction fails its condition code check.

The erratum also requires additional timing conditions to reach the point of failure, which are specific to the Cortex-A9 micro-architecture and cannot directly be controlled by software.

Projected Impact:

The erratum causes processor deadlock.

Workarounds:

The recommended workaround for this erratum is to add a DMB or DSB instruction between the write to the Strongly Ordered memory region and the conditional LDREX.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP. No conditional usage of LDREX instructions in the BSP.

ERR005386 ARM/MP: 782773—Updating a translation entry to move a page mapping might erroneously cause an unexpected translation fault

Description:

Under certain conditions specific to the Cortex-A9 micro-architecture, a write operation that updates a Cacheable translation table entry might cause both the old and the new translation entry to be temporarily invisible to translation table walks, thus erroneously causing a translation fault.

Conditions:

The erratum occurs when the following conditions are met:

- The processor has its Data Cache and MMU enabled.
- The TTB registers are set to work on Cacheable descriptors memory regions.
- The processor is updating an existing Cacheable translation table entry, and this write operation hits in the L1 Data Cache.
- A hardware translation table walk is attempted. The hardware translation table walk can be either due to an Instruction fetch, or due to any other instruction execution that requires an address translation, including any load or store operation. This hardware translation walk must attempt to access the entry being updated in condition 2, and that access must hit in the L1 Data Cache.

In practice, this scenario can happen when an operating system (OS) is changing the mapping of a physical page. The OS might have an existing mapping to a physical page (the old mapping), but wants to move the mapping to a new page (the new mapping). To do this, the OS might:

1. Write a new translation entry, without cancelling the old one. At this point the physical page is accessible using either the old mapping or the new mapping.
2. Execute a DSB instruction followed by an ISB instruction pair, to ensure that the new translation entry is fully visible.
3. Remove the old entry.

Due to the erratum, this sequence might fail because it can happen that neither the new mapping, nor the old mapping, is visible after the new entry is written, causing a Translation fault.

Projected Impact:

The erratum causes a Translation fault.

Workarounds:

The recommended workaround is to perform a clean and invalidate operation on the cache line that contains the translation entry before updating the entry, to ensure that the write operation misses in the Data Cache. This workaround prevents the micro-architectural conditions for the erratum from happening. Interrupts must be temporarily disabled so that no interrupt can be taken between the maintenance operation and the translation entry update. This avoids the possibility of the interrupt service routine bringing the cache line back in the cache.

Another possible workaround is to place the translation table entries in Non-Cacheable memory areas, but this workaround is likely to have a noticeable performance penalty.

Note that inserting a DSB instruction immediately after writing the new translation table entry significantly reduces the probability of hitting the erratum, but it is not a complete workaround.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005387 ARM/MP: 782774—A spurious event 0x63, “STREX passed,” can be reported on an LDREX that is preceded by a write to Strongly Ordered memory region

Description:

A write to Strongly Ordered memory region, followed by the execution of an LDREX instruction, can cause the “STREX passed” event to be signaled even if no STREX instruction is executed.

As a result, the event 0x63 count might be faulty, reporting too many “STREX passed” events.

This erratum also affects the associated PMUEVENT[27] signal. This signal will report the same spurious events.

Conditions:

The erratum occurs when the following conditions are met:

- The processor executes a write instruction to a Strongly Ordered memory region.
- The processor executes an LDREX instruction.
- No DSB instruction is executed, and there is no exception call or exception return, between the write and the STREX instructions.

Under these conditions, if the write instruction to Strongly Ordered memory region receives its acknowledge (BRESP response on AXI) while the LDREX is being executed, the erratum can happen.

Projected Impact:

The erratum leads to a faulty count of event 0x63, or incorrect signaling of PMUEVENT[27].

Workarounds:

The workaround for this erratum is to insert a DMB or DSB instruction between the write to Strongly Ordered memory region and the LDREX instruction.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005391 ARM: Debug CTI interrupt can cause a system deadlock when power gating the core

Description:

The Cross Trigger Interface (CTI) provides an interface for trigger inputs and outputs to the device-wide cross triggering system through the cross trigger matrix (CTM). The triggers are coupled to debug-centric signals associated with the ARM platform and are mapped to a CTI interrupt. Due to an issue with the implementation logic, using this CTI interrupt when power gating the core can cause a system deadlock.

Projected Impact:

System deadlock if the CTI interrupt is enabled when power gating the core.

Workarounds:

To prevent this issue from occurring, software should mask the CTI interrupt before power gating the core.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR006259 ARM: Debug/trace functions (PMU, PTM and ETB) are disabled with absence of JTAG_TCK clock after POR

Description:

When JTAG_TCK is not toggling after power-on reset (POR), the ARM PMU, PTM, and ETB stay in their disabled states so various debug and trace functions are not available.

Projected Impact:

Limited debug/trace capability

Workarounds:

Provide at least 4 JTAG_TCK clock cycles following POR if the PMU, PTM and ETB functions will be used. A free-running JTAG_TCK can also be used.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR007007 ARM/MP: 794073 -- Speculative instruction fetches with MMU disabled might not comply with architectural requirements

Description:

When the MMU is disabled, the ARM processor must follow some architectural rules regarding speculative fetches and the addresses to whichthese can be initiated. These rules avoid potential read accesses to read-sensitive areas. For more information about these rules, see the description of “Behavior of instruction fetches when all associated MMUs are disabled” in the *ARM Architecture Reference Manual*, ARMv7-A and ARMv7-R edition.

A Cortex-A9 processor usually operates with both the MMU and branch prediction enabled. If the processor operates in this condition for any significant amount of time, the BTAC (branch target address cache) will contain branch predictions. If the MMU is then disabled, but branch prediction remains enabled, these stale BTAC entries can cause the processor to violate the rules for speculative fetches.

The erratum can occur only if the following sequence of conditions is met:

1. MMU and branch prediction are enabled.
2. Branches are executed.
3. MMU is disabled, and branch prediction remains enabled.

Projected Impact:

If the above conditions occur, it is possible that, after the MMU is disabled, speculative instruction fetches might occur to read-sensitive locations.

Workarounds:

The recommended workaround is to invalidate all entries in the BTAC, by executing a BPIALL operation (invalidate entire branch prediction array) followed by a DSB, before disabling the MMU.

Another possible workaround is to disable branch prediction when disabling the MMU, and keep branch prediction disabled until the MMU is re-enabled.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround to be applied in a future BSP release.

ERR007008 ARM/MP: 794074 --A write request to Uncacheable Shareable memory region might be executed twice

Description:

Under certain timing circumstances specific to the Cortex-A9 microarchitecture, a write request to an Uncacheable, Shareable, Normal memory region might be executed twice, causing the write request to be sent twice on the AXI bus. This might happen when the write request is followed by another write into the same naturally aligned doubleword memory region, without a DMB between the two writes.

The repetition of the write usually has no impact on the overall behavior of the system, unless the repeated write is used for synchronization purposes.

The erratum requires the following conditions:

- A write request is performed to an Uncacheable, Shareable, Normal memory region.
- Another write request is performed into the same naturally doubleword-aligned memory region. This second write request must not be performed to the exact same bytes as the first store.

A write request to Normal memory region is treated as Uncacheable in the following cases:

1. The write request occurs while the data cache is disabled.
2. The write request is targeting a memory region marked as Normal Memory Non-Cacheable or Cacheable Write-Through.
3. The write request is targeting a memory region marked as Normal Memory Cacheable Write-Back and Shareable, and the CPU is in AMP mode.

Projected Impact:

This erratum might have implications in a multimaster system where control information is passed between several processing elements in memory using a communication variable, for example a semaphore. In such a system, it is common for communication variables to be claimed using a Load-Exclusive/Store-Exclusive, but for the communication variable to be cleared using a non-Exclusive store. This erratum means that the clearing of such a communication variable might occur twice. This might lead to two masters apparently claiming a communication variable, and therefore might cause data corruption to shared data.

Here is a scenario in which this might happen:

```
MOV r1,#0x40                ; address is double-word aligned, mapped in normal noncacheable
                              shareable memory
Loop: LDREX r5, [r1,#0x0]   ; read the communication variable
CMP r5, #0                  ; check if 0
STREXEQ r5, r0, [r1]        ; attempt to store new value
CMPEQ r5, #0                ; test if store succeeded
BNE Loop                    ; retry if not
DMB                         ; ensures that all subsequent accesses are observed when gaining
                              of the communication variable has been observed
                            ; loads and stores in the critical region can now be performed
MOV r2,#0
MOV r0, #0
```

```
DMB                          ; ensure all previous accesses are observed before the
                               communication variable is cleared
STR r0, [r1]                 ; clear the communication variable with normal store
STR r2, [r1,#0x4]            ; previous STR might merge and be sent again, which might cause
                               undesired release of the communication variable.
```

This scenario is valid when the communication variable is a byte, a half-word, or a word.

Workarounds:

There are several possible workarounds:

1. Add a DMB after clearing a communication variable:

   STR r0, [r1] ; clear the communication variable

   DMB ; ensure the previous STR is complete

   Also, any IRQ or FIQ handler must execute a DMB at the start to ensure the clearing of any communication variable is complete.
2. Ensure there is no other data using the same naturally aligned 64-bit memory location as the communication variable:

   ALIGN 64

   communication_variable DCD 0

   unused_data DCD 0

   LDR r1,= communication_variable
3. Use a Store-Exclusive to clear the communication variable, rather than a non-Exclusive store.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround to be applied in a future BSP release.

ERR005311 CCM: After exit from WAIT mode, unwanted interrupt(s) taken during WAIT mode entry process could cause cache memory corruption

Description:

An issue can occur when the ARM core is awakened very shortly after entering the WFI.

When WAIT mode is entered, the assertion of the Deep Sleep signal to the Platform memories is delayed and occurs before the clocks to the ARM core are disabled.

If an interrupt arrives immediately after executing WFI, the core will resume execution before the clocks are stopped, and it might access cache platform memories, which are already in deep sleep mode.

Projected Impact:

Software might access corrupted cache content after exit from WAIT mode and cause a system failure.

Workarounds:

To prevent this issue from occurring, software should ensure that the ARM to IPG clock ratio is less than 12:5 (that is < 2.4x), before entering WAIT mode. For example, if the IPG clock frequency is configured at 66 MHz, configuring ARM clock frequency at 142 MHz, or higher, before entering WAIT mode, is a valid software workaround.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER3

ERR006223 CCM: Failure to resume from Wait/Stop mode with power gating

Description:

When entering Wait/Stop mode with power gating of the ARM core(s), if an interrupt arrives during the power-down sequence, the system could enter an unexpected state and fail to resume.

Projected Impact:

Device might fail to resume from low-power state.

Workarounds:

Use REG_BYPASS_COUNTER (RBC) to hold off interrupts when the PGC unit is in the middle of the power-down sequence. The counter needs to be set/cleared only when there are no interrupts pending. The counter needs to be enabled as close to the WFI (Wait For Interrupt) state as possible.

The following equation can be used to aid determination of the RBC counter value:

RBC_COUNT × (1/32K RTC Frequency) ≥ (25 + PDNSCR_SW2ISO) × (1/IPG_CLK Frequency)

PDNSCR_ISO2SW = PDNSCR_ISO = 1 (counts in IPG_CLK clock domain)

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Implemented in BSP version GA L3.0.35_12.10.02_GA

ERR007265 CCM: When improper low-power sequence is used, the SoC enters low power mode before the ARM core executes WFI

Description:

When software tries to enter Low-Power mode with the following sequence, the SoC enters Low-Power mode before the ARM core executes the WFI instruction:

1. Set CCM_CLPCR[1:0] to 2’b00
2. ARM core enters WFI
3. ARM core wakeup from an interrupt event, which is masked by GPC or not visible to GPC, such as an interrupt from a local timer
4. Set CCM_CLPCR[1:0] to 2’b01 or 2’b10
5. ARM core executes WFI

Before the last step, the SoC enters WAIT mode if CCM_CLPCR[1:0] is set to 2’b01, or STOP mode if CCM_CLPCR[1:0] is set to 2’b10.

Projected Impact:

This issue can lead to errors ranging from module underrun errors to system hangs, depending on the specific use case.

Workarounds:

Software workaround:

1) Software should trigger IRQ #32 (IOMUX) to be always pending by setting IOMUX_GPR1_GINT

2) Software should then unmask IRQ #32 in GPC before setting CCM Low-Power mode

3) Software should mask IRQ #32 right after CCM Low-Power mode is set (set bits 0–1 of CCM_CLPCR)

Proposed Solution:

No fix scheduled

Linux BSP Status:

A patch is included in both BSP kernels v3.10.9 and v3.0.35.

ERR006287 eLCDIF/EPDC/PXP/SPDC: Display domain modules register read may return value for the prior read access after resuming from power gating

Description:

Upon resuming from power gating, the modules in the display power domain (eLCDIF , EPDC, PXP and SPDC) might fail to perform register reads correctly.

Projected Impact:

Power gating is not available on the display power domain, if the modules listed above are used.

Workarounds:

When the modules listed above are used, do not use power gating on the display power domain. Display power domain power-down is controlled by display_pdn_req in the GPC_CNTR register.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Implemented in BSP version GA L3.0.35_12.10.02_GA

ERR004573 EPDC: Collision status must be read before clearing IRQ

Description:

When an interrupt event occurs, the appropriate bit within the EPDC_IRQ register is set by the EPDC. Once the interrupt has been handled, software clears the interrupt source by writing to the CLEAR address. However, when the EPDC interrupt is cleared, the Look Up Table (LUT) status of LUTs, 16–63, will be cleared and lost before software can capture it.

Projected Impact:

Collision status of LUTs, 16–63, gets cleared incorrectly, when EPDC interrupt is cleared.

Workarounds:

When handling an interrupt, the software must capture the LUT status by reading the EPDC_STATUS_LUTS1 and EPDC_STATUS_LUTS2 registers, before clearing the interrupt.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER3

ERR005313 EPDC: Incorrect data fetched when the buffer update width is 2048 pixels or greater

Description:

When sending an image update to a high resolution panel with an image buffer width ≥ 2048 pixels, the EPDC module fetches incorrect data because an internal address register is incorrectly truncated.

Projected Impact:

EPDC might fetch incorrect data. The INIT update with FIXNP is not affected by this issue.

Workarounds:

Split the update down to < 2048 pixels. If group updates are supported, send updates in a group, so all of them get scanned together exactly like a single update.

If the group update feature is not available, send updates sequentially (one after another immediately after Working Buffer (WB) is done, no need to wait for FRAME scan to complete). One frame difference on scan time is the theoretical worst case; however, this was not visually observed during testing by Freescale.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version GA L3.0.35_4.0.0

ERR004365 EXSC: Exclusive accesses to certain memories are not supported to full AXI specification

Description:

Any exclusive operation to PSRAM or other RAM type's connected to the EIM returns an incorrect response of "EXOKAY", indicating that exclusive writes are always successful.

Projected Impact:

EIM does not support exclusive accesses according to AXI specifications.

Workarounds:

Use DDR or OCRAM memories when performing exclusive accesses.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005828 EXSC: Protecting the EIM memory map region causes unpredictable behavior

Description:

If a write access to the EIM address region is denied due to the CSU access control policy, then, all subsequent write accesses to the EIM region will write unintended data.

Projected Impact:

Blocking write accesses to the EIM region through Trustzone is not supported.

Workarounds:

To prevent unpredictable behavior, prior to accessing the EIM region, set all bits in the EIM CSU_CSL field to 1 so that all accesses are allowed. Specifically:

EIM: CSU_CSL31[23:16] = 0xff

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP. The BSP does not use CSU.

ERR005908 GPU2D: Image quality degradation observed for stretch blits when the stretch factor is exactly an integer

Description:

GPU2D supports BLIT acceleration by using the Graphics Device Interface (GDI) API. When using the stretch blit GDI API, if the stretch factor is exactly an integer, the resulting image has rendering errors.

Projected Impact:

Minor visual impact in image quality when using the stretch blit for hardware acceleration. The rendering result using the BLIT hardware acceleration will not be the same as the software rendered image.

Workarounds:

There are no software workarounds that completely resolve the issue. The filter blit API can be used instead of the stretch blit for BLIT acceleration; however, there will be a performance impact and might not be suitable for all applications. The issue is not observed when the stretch blit for BLIT acceleration is not used.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR004307 I/O: USB_HSIC interface should not be configured to Differential input mode

Description:

DDR3, LPDDR2, and USB_HSIC interfaces are of the DDR I/O type, thus having the option to work in DDR input mode. This mode requires setting the DRAM_VREF to half the I/O voltage. This reference pad is used in all DDR type I/O interfaces. Since all I/O interfaces do not have a common voltage, configuring more than two I/O interfaces to DDR input mode might not work.

Conditions:

De-assertion of POR_B when the SoC is powered-up.

Projected Impact:

DDR3, LPDDR2, and USB_HSIC interfaces might not work together.

Workarounds:

Configure the DDR_INPUT bit in IOMUXC for USB_HSIC to “0,” that is, CMOS input type.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP. The BSP ensures that the DDR_INPUT bit is set to CMOS input type.

ERR007805 I2C: When the I2C clock speed is configured for 400 kHz, the SCL low period violates the I2C specification

Description:

When the I2C module is programmed to operate at the maximum clock speed of 400 kHz (as defined by the I2C spec), the SCL clock low period violates the I2C spec of 1.3 uS min. The user needs to reduce the clock speed to get the SCL low time to meet the 1.3us I2C minimum required. This behavior means the SoC is not compliant to the I2C spec at 400 kHz.

Projected Impact:

No failures have been observed when operating at 400 kHz. This erratum only represents a violation of the I2C specification for the SCL low period.

Workarounds:

In order to exactly meet the clock low period requirement at fast speed mode, SCL must be configured to 384 KHz or less.

The following clock configuration meets the I2C specification requirements for SCL low for i.MX 6 products:

- I2C parent clock PERCLK_ROOT = 24 M OSC
- perclk_podf = 1
- PERCLK_ROOT = 24M OSC/perclk_podf = 24 MHz
- I2C_IFDR = 0x2A
- I2C clock frequency = 24 MHz/64 = 375 kHz

Proposed Solution:

No fix scheduled

Linux BSP Status:

The BSP configures the I2C frequency to 375 kHz by default.

ERR005778 MMDC: DDR Controller's measure unit may return an incorrect value when operating below 100 MHz

Description:

The measure unit counts cycles of an internal ring oscillator. The measure unit readout is used to fine tune the delay lines for temperature/voltage changes for both DDR3 and LPDDR2 interfaces. When operating at low frequencies (below 100 MHz), the measure unit counter might overflow due to an issue in the overflow protection logic. As a result, an incorrect measure value will be read.

Projected Impact:

This might cause a rare issue if the measure unit counter stops within a small range of values that translate to a delay that tunes the system incorrectly. This issue might not manifest in the application because it is dependent on a combination of DDR frequencies coupled with specific Process, Voltage, and Temperature conditions.

Workarounds:

To workaround this issue, following steps should be performed by software:

1. Prior to reducing the DDR frequency (400 MHz), read the measure unit count bits (MU_UNIT_DEL_NUM).
2. Bypass the automatic measure unit when below 100 MHz, by setting the measure unit bypass enable bit (MU_BYP_EN).
3. Double the measure unit count value read in step 1 and program it in the measure unit bypass bit (MU_BYP_VAL) of the MMDC PHY Measure Unit Register, for the reduced frequency operation below 100 MHz.

Software should re-enable the measure unit when operating at the higher frequencies, by clearing the measure unit bypass enable bit (MU_BYP_EN). This code should be executed out of Internal RAM or a non-DDR based external memory.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER4

ERR007927 ROM: 32 kHz internal oscillator timing inaccuracy may affect SD/MMC and OneNAND boot

Description:

The internal boot ROM uses the general-purpose timer (GPT) as a timing reference for event and timeout measurement during the boot process. The ROM uses the 32 kHz clock as the clock source for the GPT. There will be a short period during device power-up when the SoC will be using the internal ring oscillator until the crystal oscillator is running. Once the crystal oscillator is running, the SoC will automatically switch from the internal oscillator to the crystal oscillator. Consequently, there will be a period of time when the SoC will be booting and using the internal ring oscillator as its reference clock and the ROM code will be dependent on that clock.

The internal ring oscillator is less accurate than a crystal oscillator and may be up to two times faster than a 32 kHz external crystal oscillator. The ROM code assumes the reference clock is 32 kHz, so in the presence of a faster reference clock some delays or timeout configurations in the ROM code will be shorter than expected and may affect SD/MMC boot and One NAND boot.

NOR and SPI-NOR boot modes are not affected by this issue because these modes do not use timeouts.

The potential effects are:

1. The SD/MMC card specification may be violated if the SD/MMC card Nac parameter is larger than 50 ms, or if its initialization time is greater than 500 ms.
2. According to the SD 3.0 specification, the controller should wait a minimum of 5 ms after disabling SDCLK before re-enabling SDCLK when voltage switching. In the worst case, the ROM code may only wait 2.5 ms.
3. According to the SD 3.0 specification, the timeout for a CMD6 data transaction response is 100 ms. In the worst case, the ROM code may timeout after 50 ms and therefore not conform to the specification.
4. One NAND boot may fail if the One NAND memory tRD1 is greater than 1.5 ms.

Projected Impact:

To date, this failure has not been observed on any system. The description and workarounds presented here are based on analysis of the timings in the ROM boot sequence and indicates the possibility that SD/MMC boot or OneNAND boot may be affected. SD/MMC card specifications may not be met during boot.

Workarounds:

SDMMC boot:

1. SD/MMC: Choose an SD/MMC card for which the Nac parameter is to be specified less than 50 ms and its initialization time is less than 500 ms.

2. Choose the “SD/SDXC Speed” SDR12/SDR25 fuse configuration instead of SDR50/SDR104 when booting from an SD 3.0 card. SDR12/SDR25 is the default configuration. See i.MX6 device reference manual Fuse Map chapter for details on these fuses. If SD Card operation at a higher speed is desired, the SD/MMC can be reconfigured after ROM boot. Note that these fuses are also affected by ERR005645.
3. Boot from SPI-NOR initially then switch to SD/MMC or One NAND once the external 32 kHz clock is stable.
4. Extend the assertion of POR_B until the 32 kHz crystal oscillator is running and stable.
5. Provide an external stable 32 kHz clock input prior to de-assertion of POR_B.

OneNAND boot:

1. OneNAND: Choose a OneNAND memory with tRD1 less than 1.5 ms.
2. Boot from SPI-NOR initially then switch to SD/MMC or One NAND once the external 32 kHz clock is stable.
3. Extend the assertion of POR_B until the 32 kHz crystal oscillator is running and stable.
4. Provide an external stable 32 kHz clock input prior to de-assertion of POR_B.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR005645 ROM: Normal SD clock speed (SDR12) not selectable in SD/SDXC boot mode

Description:

When booting in SD/SDXC boot mode, users cannot set the SD clock speed to Normal mode (SDR12). Selecting the SDR12 boot switch setting for BOOT_CFG1[3:2] in the fuse table will default to the High Speed mode (SDR25) due to an incorrect mapping in the boot ROM.

Projected Impact:

Due to this mapping issue, the user cannot select Normal SD clock speed at boot time; however, this will not cause any issues. For an older SD device that does not support high-speed mode, ROM will first attempt high speed mode and when this mode fails will automatically switch to normal speed and continue normally with the boot process. This mapping issue does not impact MMC boot.

Workarounds:

None. The minimum SD clock speed supported is high-speed mode (SDR25) for initial booting in SD/SDXC boot mode. When booting with an SD card that only supports SD clock speed in Normal mode (SDR12), users need to be aware of the revised SD/SDXC boot mode switch settings for BOOT_CFG1[3:2] given in Table 3. The automatic switch from high speed to normal speed is transparent to the user.

Table 3. Revised SD/SDXC Boot Mode Switch Settings for BOOT_CFG1[3:2]

BOOT_CFG1[3:2]	SD/SDXC Boot Speed
0x	SDR25
10	SDR50
11	SDR104

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005768 ROM: In rare cases, secondary image boot flow may not work due to mis-sampling of the WDOG reset

Description:

In case the primary image authentication fails, ROM will try to perform a WDOG reset and boot with the secondary image. However ROM does not set the SRE bit of watchdog control register which might cause a WDOG reset failure occasionally and result in ROM staying in an endless loop.

Projected Impact:

The secondary boot might not work in the first attempt.

Workarounds:

There are no software workarounds for this issue, instead the user will need to reboot the IC, which will force a second iteration of the secondary boot.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR003778 SSI: In AC97, 16-bit mode, received data is shifted by 4-bit locations

Description:

When the SSI is configured in AC97, 16-bit mode, the Rx data is received in bits [19:4] of the RxFIFO, instead of [15:0] bits.

Projected Impact:

The SDMA script should be updated accordingly to perform the shift to the right location on the fly during data transfer. If the data register is accessed directly by software, it should account for the shifted data and perform shifting to the right location.

Workarounds:

The data should be shifted to the right location by the SDMA script or by the software in case of direct access to the register.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP.

ERR006281 USB: Incorrect DP/DN state when only VBUS is applied

Description:

When VBUS is applied without any other supplies, incorrect communication states are possible on the data (DP/DN) signals. If VDDHIGH_IN is supplied, the problem is removed.

Projected Impact:

This issue primarily impacts applications using charger detection to signal power modes to a PMIC in an undercharged battery scenario where the standard USB current allotment is not sufficient to boot the system.

Workarounds:

Apply VDDHIGH_IN if battery charge detection is needed. Otherwise, disable charger detection by setting the EN_B bit in USB_ANALOG_USBx_CHRG_DETECTn to 1.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR006308 USB: Host non-doubleword –aligned buffer address can cause host to hang on OUT Retry

Description:

The USB host core operating in streaming mode might underrun while sending the data packet of an OUT transaction. The host then retries the OUT transaction according to the USB specification.

This issue occurs during the OUT retry. The USB host might hang on OUT retry if the data buffer start address is not 4-byte aligned.

This applies to both the host controller and the OTG controller in host mode.

Projected Impact:

Host controller only transmits SOF packets. All other traffic is blocked.

Workarounds:

- Set the host TXFIFO threshold to a large value (TXFIFOTHRES in the TXFILLTUNING register). This increases the tolerance to bus latency and avoids a FIFO underrun.
- Set the Stream Disable bit (SDIS) to 1 in the USBMODE register. This forces the controller to load an entire packet in the FIFO before starting to transmit on the USB bus. Hence, the FIFO never underruns. This somewhat reduces the maximum bandwidth of the USB, because there is idle time when the the controller waits for the entire packet to be loaded.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround implemented in Linux BSP version L3.0.35_4.0.0

ERR004536 uSDHC: ADMA Length Mismatch Error occurs if the AHB read access is slow, when reading data from the card

Description:

When uSDHC reads data from an external device in ADMA1/ADMA2 mode, the block counter in the ADMA module will be updated after the ADMA reads the description from memory through AHB bus. If the uSDHC has finished one block read from external device before the ADMA read description from memory is done, the updated block counter in the ADMA is not correct. An ADMA Length Mismatch Error will occur.

The issue is triggered based on two conditions:

- The latency of AHB read access (t1)
- The interval from the time when read command is issued to the time when the first block read is done (t2). If t1*N > t2, then this issue will occur here, N=1 for ADMA1 mode because only one AHB read access is needed to fetch the descriptor in ADMA1 mode; N=2 for ADMA2 mode because only two AHB read accesses are needed to fetch the descriptor in ADMA2 mode.

For SD card or eMMC applications, ADMA mode should not cause this issue because the block size is 512 bytes for SD card and eMMC applications. One block read takes more than 2.8 μs for eMMC4.5, more than 5.4 μs for SD3.0 card, and much more time for legacy eMMC and SD cards.

Projected Impact:

If the total latency of these two (or one) AHB SINGLE accesses is longer than the time to read one block from the card, the incorrect block count will be loaded by the DMA engine.

Workarounds:

For SDIO2.0/SDIO3.0 cards, whose minimum block size can be 4 bytes, ADMA modes should not be used. Use SDMA (or ADMA1) in case the AHB latency is larger than the minimal time for one block.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in BSP version ER3. SDMA mode is used instead of ADMA mode.

ERR006282 ROM code uses nonreset PFDs to generate clocks, which may lead to random boot failures

Description:

The phase fractional dividers (PFDs) can go into an unknown state if they are not properly reset before being used as clock sources. There are two outcomes to this failure:

- PFD in an unknown state—This outcome affects the boot sources within the chip (for example, eMMC, NAND):
  - The ROM boot code fails to properly reset the PFDs, which are used for the bootable sources within the processor (for example, NOR, SD, eMMC). This has the potential of putting the PFDs into an unknown state in rare circumstances where the boot source IP blocks do not receive correct clocks and the chip subsequently fails to boot.
  - This failure has only been observed when the processor is subjected to boot 'stress testing' whereby the processor is subjected to back-to-back reboots. This failure only affects a small subset of processors.
  - The number of back-to-back reboots required to see the issue varies but has ranged from hundreds to thousands of back-to-back reboots on that subset of parts. Additionally, the error has been observed to not have any "memory," in that encountering the error does not make it more likely to encounter it again on the next boot.
- Resume from Suspend when PLL is bypassed—This outcome can also cause the PFDs to lose state if the PLL is configured in BYPASS mode and not reset correctly. If the PLL is not BYPASSED, then the suspend/resume functionality is not affected. This issue is less likely to be observed with PLL3 on i.MX6 Dual/Quad than the PLLs on i.MX6 SoloLite because the PLLs have a faster lock time, thereby reducing the possibility of the PFD state loss.

Projected Impact:

All boot modes are affected by this issue since all boot devices rely on a PFD-sourced clock tree.

For the potential Resume-from-Suspend failure, the u-boot patch in the L3.0.35_2.1.0 Linux BSP prevents this failure.

ENGR00236856 - Reset PFDs when enabling PLL.patch

Workarounds:

None

Proposed Solution:

Fixed in i.MX6SL silicon revision 1.2.

Linux BSP Status:

No software workaround for all boot devices.

U-boot patch with the correct procedure to reset the PFDs will be available on the L3.0.35_2.1.0 Linux BSP.

ERR007266 ROM: EIM NOR boot may fail if plug-in is used

Description:

The issue occurs when the two conditions below are both met:

- EIM NOR boot with plug-in is used, and
- Plug-in was specified running in the on-chip RAM (OCRAM).

The ROM sets 0x907000 as the initial address of the source image (0x8000000 was expected) after pu_irom_hwcnfg_setup is called. The problem occurs when the plug-in calls this function again.

Projected Impact:

EIM NOR boot might fail if the workaround is not applied.

Workarounds:

There are two workarounds for this issue:

- Modify the initial address to 0x8000000 (EIM nor base address) in the plug-in before pu_irom_hwcnfg_setup is called, or
- Specify the plug-in runs in EIM NOR directly instead of in internal RAM

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround in u-boot v2013.04 and in L3.10.9_1.0.0_alpha release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: IMX6SLCE
Rev. 4
07/2014