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.

For details on part marking and silicon revision level identification and comparison, see the "Ordering Information" section of the *i.MX 6SoloLite Applications Processor Data Sheet* for your device.

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 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Changes
Rev. 5	02/2019	***Added the following errata:*** ERR004535 ERR007881 ERR008990 ERR009165 ERR009219 ERR009535 ERR009541 ERR009596 ERR009604 ERR009606 ERR009678 ERR009742 ERR009743 ERR009858 ERR010481 ERR010822 ERR011121 ERR011421 ERR050070 **Updated the following errata:** ERR003717 ERR003718 ERR003719 ERR003721 ERR003723 ERR003724 ERR003725 ERR003726 ERR003727 ERR003728 ERR003729 ERR003730 ERR003731 ERR003732 ERR003733 ERR003734 ERR003735 ERR003736 ERR003737 ERR003738 ERR003739 ERR003741 ERR003743 ERR003778 ERR004307 ERR004326 ERR004327 ERR004536 ERR004573 ERR005175 ERR005183 ERR005185 ERR005187 ERR005198 ERR005313 ERR005382 ERR005383 ERR005385 ERR005386 ERR005387 ERR005391 ERR005645 ERR005778 ERR005828 ERR005852 ERR005908 ERR006223 ERR006259 ERR006281 ERR007007 ERR007008 ERR007265 ERR007266 ERR007805 ERR009219 ***Updated***: Table 2, Document Revision History, ordered numerically, removed redundant items, and tabulated. ***Removed the following:*** ERR005777 *Removed*: Figure 1, Revision Level to Part Marking Cross-Reference. Added reference to Data Sheet.
Rev. 4	06/2014	• Added the following: ERR007805 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.

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	8
ARM®			
ERR003717	ARM: 740657—Global Timer can send two interrupts for the same event	No fix scheduled	9
ERR003718	ARM: 743622—Faulty logic in the Store Buffer may lead to data corruption	No fix scheduled	11
ERR003719	ARM: 751469—Overflow in PMU counters may not be detected	No fix scheduled	13
ERR003721	ARM: 751473—Under very rare circumstances, Automatic Data prefetcher can lead to deadlock or data corruption	No fix scheduled	15
ERR003723	ARM: 751476—May miss a watchpoint on the second part of an unaligned access that crosses a page boundary	No fix scheduled	16
ERR003724	ARM: 754322—Possible faulty MMU translations following an ASID switch	No fix scheduled	17
ERR003725	ARM: 725631—ISB is counted in Performance Monitor events 0x0C and 0x0D	No fix scheduled	19
ERR003726	ARM: 729817—MainID register alias addresses are not mapped on Debug APB interface	No fix scheduled	20
ERR003727	ARM: 729818—In debug state, next instruction is stalled when sdabort flag is set, instead of being discarded	No fix scheduled	21
ERR003728	ARM: 740661—Event 0x74 / PMUEVENT[38:37] may be inaccurate	No fix scheduled	22
ERR003729	ARM: 740663—Event 0x68 / PMUEVENT[9:8] may be inaccurate	No fix scheduled	23
ERR003730	ARM: 743623—Bad interaction between a minimum of seven PLDs and one Non-Cacheable LDM can lead to a deadlock	No fix scheduled	25
ERR003731	ARM: 743626—An imprecise external abort, received while the processor enters WFI, may cause a processor deadlock	No fix scheduled	27
ERR003732	ARM: 751471—DBGPCSR format is incorrect	No fix scheduled	28
ERR003733	ARM: 751480—Conditional failed LDREXcc can set the exclusive monitor	No fix scheduled	30
ERR003734	ARM: 752519—An imprecise abort may be reported twice on non-cacheable reads	No fix scheduled	31
ERR003735	ARM: 754323—Repeated Store in the same cache line may delay the visibility of the Store	No fix scheduled	32
ERR003736	ARM: 756421—Sticky Pipeline Advance bit cannot be cleared from debug APB accesses	No fix scheduled	34
ERR003737	ARM: 757119—Some “Unallocated memory hint” instructions generate an UNDEFINED exception instead of being treated as NOP	No fix scheduled	35
ERR003738	ARM: 751475—Parity error may not be reported on full cache line access (eviction / coherent data transfer / cp15 clean operations)	No fix scheduled	36
ERR003739	ARM: 751470—Imprecise abort on the last data of a cache linefill may not be detected	No fix scheduled	37

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	38
ERR003743	ARM/PL310: 754670—A continuous write flow can stall a read targeting the same memory area	No fix scheduled	39
ERR004326	ARM/MP: 761321—MRC and MCR are not counted in event 0x68	No fix scheduled	40
ERR004327	ARM/MP: 764319—Read accesses to DBGPRSR and DBGOSLSR may generate an unexpected UNDEF	No fix scheduled	41
ERR005175	ARM/MP: 771221—PLD instructions may allocate data in the Data Cache regardless of the Cache Enable bit value	No fix scheduled	42
ERR005183	ARM/MP: 771224—Visibility of Debug Enable access rights to enable/disable tracing is not ensured by an ISB	No fix scheduled	43
ERR005185	ARM/MP: 771225—Speculative cacheable reads to aborting memory region clear the internal exclusive monitor, may lead to livelock	No fix scheduled	44
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	46
ERR005198	ARM/PL310: 780370—DATAERR, TAGERR, and Tag parity errors are incorrectly sampled by the eviction buffer, leading to data corruption	No fix scheduled	47
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	50
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	51
ERR005385	ARM/MP: 782772—A speculative execution of a Load-Exclusive or Store-Exclusive instruction after a write to Strongly Ordered memory might deadlock the processor	No fix scheduled	52
ERR005386	ARM/MP: 782773—Updating a translation entry to move a page mapping might erroneously cause an unexpected translation fault	No fix scheduled	54
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	56
ERR005391	ARM: Debug CTI interrupt can cause a system deadlock when power gating the core	No fix scheduled	57
ERR006259	ARM: Debug/trace functions (PMU, PTM and ETB) are disabled with absence of JTAG_TCK clock after POR	No fix scheduled	58
ERR007007	ARM/MP: 794073—Speculative instruction fetches with MMU disabled might not comply with architectural requirements	No fix scheduled	59
ERR007008	ARM/MP: 794074—A write request to Uncacheable Shareable memory region might be executed twice	No fix scheduled	60
ERR009604	ARM (CA9): 845369—Under very rare timing circumstances, transition into streaming mode might create a data corruption	No fix scheduled	62
ERR009742	ARM: 795769—“Write Context ID" event is updated on read access	No fix scheduled	64
ERR009743	ARM: 799770—DBGPRSR Sticky Reset status bit is set to 1 by the CPU debug reset instead of by the CPU non-debug reset	No fix scheduled	65

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ERR009858	ARM/PL310: 796171—When data banking is implemented, data parity errors can be incorrectly generated	No fix scheduled	66
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	67
ERR006223	CCM: Failure to resume from Wait/Stop mode with power gating	No fix scheduled	68
ERR007265	CCM: When improper low-power sequence is used, the SoC enters low power mode before the ARM core executes WFI	No fix scheduled	69
ERR009219	CCM: Asynchronous clock switching can cause unpredictable behavior	No fix scheduled	70
DCIC			
ERR011100	DCIC: The External Controller Mismatch Indication Signal Outputs Are Not Usable	No fix scheduled	71
eCSPI			
ERR009165	eCSPI: TXFIFO empty flag glitch can cause the current FIFO transfer to be sent twice	No fix scheduled	72
ERR009535	eCSPI: Burst completion by SS signal in slave mode is not functional	No fix scheduled	73
ERR009606	eCSPI: In master mode, burst lengths of 32n+1 will transmit incorrect data	No fix scheduled	74
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	75
EPDC			
ERR004573	EPDC: Collision status must be read before clearing IRQ	No fix scheduled	76
ERR005313	EPDC: Incorrect data fetched when the buffer update width is 2048 pixels or greater	No fix scheduled	77
EXSC			
ERR004365	EXSC: Exclusive accesses to certain memories are not supported to full AXI specification	No fix scheduled	78
ERR005828	EXSC: Protecting the EIM memory map region causes unpredictable behavior	No fix scheduled	79
GPU			
ERR005908	GPU2D: Image quality degradation observed for stretch blits when the stretch factor is exactly an integer	No fix scheduled	80
I/O			
ERR004307	I/O: USB_HSIC interface should not be configured to Differential input mode	No Fix scheduled	81
I2C			
ERR007805	I2C: When ERR004307 I2C clock speed is configured for 400 kHz, the SCL low period violates the I2C specification	No fix scheduled	82

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
MMDC			
ERR005778	MMDC: DDR Controller's measure unit may return an incorrect value when operating below 100 MHz	No fix scheduled	83
ERR009596	MMDC: ARCR_GUARD bits of MMDC Core AXI Re-ordering Control register (MMDC_MAARCR) doesn't behave as expected	No fix scheduled	84
ERR010481	MMDC: ZQ calibration issue when interfacing to LPDDR2/LPDDR3 memory with two chip selects	No fix scheduled	85
ERR011421	MMDC: DDR I/O glitches on power-up	No fix scheduled	87
ERR050070	MMDC: Hardware Write Leveling Calibration Error bits MMDC_MPWLGCR[WL_HW_ERRn] are incorrectly de-asserted	No fix scheduled	88
PXP			
ERR009541	PXP:CSC2 does not perform RGB to YCbCr and RGB to YUV conversions	No fix scheduled	89
ROM			
ERR005645	ROM: Normal SD clock speed (SDR12) not selectable in SD/SDXC boot mode	No fix scheduled	90
ERR005768	ROM: In rare cases, secondary image boot flow may not work due to mis-sampling of the WDOG reset	No fix scheduled	91
ERR006282	ROM code uses nonreset PFDs to generate clocks, which may lead to random boot failures	Fixed in silicon revision 1.2.	92
ERR007266	ROM: EIM NOR boot may fail if plug-in is used	No fix scheduled	93
ERR007927	ROM: 32 kHz internal oscillator timing inaccuracy may affect SD/MMC and OneNAND boot	No fix scheduled	94
ERR009678	ROM: SD/EMMC/NAND prematurely times out during boot	No fix scheduled	96
SSI			
ERR003778	SSI: In AC97, 16-bit mode, received data is shifted by 4-bit locations	No fix scheduled	97
ERR008990	SSI: Channel swap in single FIFO mode when an underrun or overrun occurs	No fix scheduled	98
System Boot			
ERR011121	System Boot: EIM NOR boot failed on closed part if image targeted into OCRAM	No fix scheduled	99
USB			
ERR004535	USB: USB suspend and resume flow clarifications	No fix scheduled	100
ERR006281	USB: Incorrect DP/DN state when only VBUS is applied	No fix scheduled	101
ERR006308	USB: Host non-doubleword –aligned buffer address can cause host to hang on OUT Retry	No fix scheduled	102
ERR007881	USB: Timeout error in Device mode	No fix scheduled	103
ERR010822	USB: USB host may not respond to RX data	No fix scheduled	104

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
	uSDHC		
ERR004536	uSDHC: ADMA Length Mismatch Error may occur for longer read latencies	Fixed in silicon revision 1.3.	105

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 Linux BSP codebase starting in release L3.0.35_4.1.0.

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 implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The BSP does not use ARM global timer. The configuration and logic of the kernel does not make use of the Global Timer. If the Global timer is used, the workaround documented by ARM should be followed. Due to limitations of this timer specifically in low power mode operation we do not recommend the use of this 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 Linux BSP codebase (UBOOT) starting in release L3.0.35_4.1.0.

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 may 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.

It is important to notice that the Cycle counter is not affected by this erratum.

Projected Impact:

PMU overflow detection is not reliable.

Workarounds:

The main workaround for this erratum is to poll the performance counter. The maximum increment in a single cycle for a given event is 2. Therefore, polling can be infrequent as no counter can increment by more than 2^32 in fewer than 2 billion cycles.

If the main usage model for performance counters is collecting values over a long period, then polling can be used to collect values (and reset the counter) rather than waiting for an overflow to occur. Polling can be done infrequently and overflow avoided.

If the main usage model for performance counters relies on presetting the counter to some value and waiting for an overflow to occur, then polling can be used to detect when an overflow event has been missed. An overflow can be determined to have been missed if the unsigned value in the counter is less than the value preset into the counter. Again, polling can be done infrequently because of the number of cycles it would need for this check to fail. In the case that the erratum was triggered and an overflow event was missed, that counter sample can be thrown away or the true value can be reconstructed.

An alternative workaround is to configure two counters to be triggered by the same event, staggering their initial count values by 1. This will result in the rollover being triggered by at least counter.

This alternative workaround works for all Cortex-A9 events but the three following ones, due to the fact these three events can increment by 2 in a single cycle:

- 0x68 - Instructions coming out of the core renaming stage
- 0x73 - Floating-point instructions
- 0x74 - NEON instructions

For these 3 events, only the first workaround is applicable to fix the defect.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround is not implemented because this erratum will not be encountered in normal device operation. The Linux BSP does not support this optional profiling feature. Users may add support for this profiling feature as required, but should ensure the multiple errata impacting the ARM PMU are considered especially for multi-core usage.

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 not implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. 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:

A software workaround is not implemented because this erratum will not be encountered in normal device operation. The Linux BSP does not use this debug feature—the ARM workaround should be followed.

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

Description:

A microTLB entry might be corrupted following an ASID switch, possibly corrupting subsequent MMU translations.

The erratum requires execution of an explicit memory access, which might be speculative. This memory access misses in the TLB and cause a translation table walk. The erratum occurs when the translation table walk starts before the ASID switch code sequence, but completes after the ASID switch code sequence. In this case, a new entry is allocated in the microTLB for the TLB entry for this translation table walk, but corresponding to the old ASID. Because the microTLB does not record the ASID value, the new MMU translation, which should happen with the new ASID following the ASID switch, might hit this stale microTLB entry and become corrupted.

Note that there is no Trustzone Security risk because the Security state of the access is held in the microTLB, and 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 Linux BSP codebase starting in release L3.0.35_4.1.0.

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:

Software workaround is not implemented because this erratum will not be encountered in normal device operation.The Linux BSP does not support this optional profiling feature. Users may add support for this profiling feature as required, but should ensure the multiple errata impacting the ARM PMU are considered especially for multi-core usage.

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 is not implemented because this erratum will not be encountered in normal device operation.

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 is not implemented because this erratum will not be encountered in normal device operation.

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:

Software workaround is not implemented because this erratum will not be encountered in normal device operation. The Linux BSP does not support this optional profiling feature. Users may add support for this profiling feature as required, but should ensure the multiple errata impacting the ARM PMU (Performance Monitoring Unit) are considered especially for multi-core usage.

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:

Software workaround is not implemented because this erratum will never be encountered in normal device operation.The Linux BSP does not support this optional profiling feature. Users may add support for this profiling feature as required, but should ensure the multiple errata impacting the ARM PMU (Performance Monitoring Unit) are considered especially for multi-core usage.

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 implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. Users should check their custom OS and either avoid the code sequence or apply the ARM recommended workaround. The ARM recommended workaround does have a performance impact.

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:

The BSP uses correct MMU settings and does not present conditions that can cause an imprecise external abort. BSP has exception handlers for such aborts.

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 is not implemented because this erratum will never be encountered in normal device operation. Software workaround not applicable to the BSP since it is a debug feature. Users should use the ARM recommended workaround if using this debug feature in their application.

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 implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.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:

Software workaround cannot be implemented to mask or workaround this SoC issue. This erratum will result in impacted or reduced functionality as described above. Users should evaluate their specific use case and apply the recommended workaround to prevent the occurrence of this erratum. Please contact your support channel if you have any questions or concerns.

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 is not implemented because this erratum will not be encountered in normal device operation. However, the ARM Linux kernel common code has added the necessary DMB in places to ensure the visibility of the written data to any external agent.The workaround for this erratum is to insert a DMB operation after the faulty write operation in code sequences that this erratum might affect, to ensure the visibility of the written data to any external agent. The BSP does use DMBs however the specific condition or scenario is not seen in kernel code.

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:

Software workaround is not implemented because this erratum will never be encountered in normal device operation.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. Software workaround not applicable to the BSP as instruction encodings are not generated by compiler.

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:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. Parity is not supported on i.MX 6 series. See erratum ERR005187 regarding the BSP interaction with the parity interrupt.

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:

A software workaround is possible but it hasn’t been implemented in the Linux BSP yet. BSP functionality may be affected in some configurations and use cases as described above. Users should evaluate their specific use case and apply the recommended workaround to prevent the occurrence of this erratum. Please contact your support channel if you have any questions or concerns.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The bit[10] “High Priority for SO and Dev reads enable” of the PL310 Auxiliary Control Register is not enabled in 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:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.

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:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.The Linux BSP does not support this optional profiling feature. Users may add support for this profiling feature as required, but should ensure the multiple errata impacting the ARM PMU (Performance Monitoring Unit) are considered especially for multi-core usage.

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 is not implemented because this erratum will never be encountered in normal device operation. Users that require this debug feature should implement the recommended ARM workaround.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The BSP does not dynamically enable/disable data cache during run-time and thus avoids the PLD instruction with the data cache off.

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 is not implemented because this erratum will never be encountered in normal device operation. Users should use ARM recommended workaround if using this debug trace feature.

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 implemented in Linux BSP codebase in all releases. Software workaround is to enable branch prediction which is enabled by default in the BSP GA release.

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 implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. BSP ignores any assertion on the PARITYFAIL[7:6] bits by masking the ARM -GIC parity interrupt 125.

Please note that the i.MX6 does not support the parity feature and hence should not be enabled.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The erratum only affects configurations which implement the parity support option. i.MX6 parity is not supported. In the Linux implementation, the parity error detection is disabled and GIC parity interrupt 125, is masked in the BSP. The parity feature is disabled by default and should not be enabled.

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:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.The Linux BSP does not support this optional profiling feature. Users may add support for this profiling feature as required, but should ensure the multiple errata impacting the ARM PMU (Performance Monitoring Unit) are considered especially for multi-core usage.

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, adding a DSB at the beginning of the abort exception handler) is implemented in Linux BSP codebase starting in release L3.0.35_4.1.0.

ERR005385 ARM/MP: 782772—A speculative execution of a Load-Exclusive or Store-Exclusive instruction after a write to Strongly Ordered memory might deadlock the processor

Description:

Under certain timing circumstances, a processor might deadlock when the execution of a write to a Strongly Ordered memory region is followed by the speculative execution of a Load-Exclusive or a Store-Exclusive instruction that is mis-speculated.

The mis-speculation can be due to either the Load-Exclusive or Store-Exclusive instruction being conditional, and failing its condition code check, or to the Load-Exclusive or Store-Exclusive instruction being speculatively executed in the shadow of a mispredicted branch.

Conditions:

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.

The erratum requires the following conditions:

— The processor executes a write instruction to a Strongly Ordered memory region

— The processor speculatively executes a Load-Exclusive or Store-Exclusive instruction that is either:

a) A conditional instruction

b) An instruction in the shadow of a conditional branch.

— The Load-Exclusive or Store-Exclusive instruction is cancelled because the speculation was incorrect, because either:

a) The conditional Load-Exclusive or Store-Exclusive instruction failed its condition-code check

b) The conditional branch was mispredicted, so that all subsequent instructions speculatively executed must be flushed, including the Load-Exclusive or Store-Exclusive.

The erratum also requires additional timing conditions to be met. These are specific to each platform, and are not controllable by software. These timing conditions includes the fact that the response to the Strongly Ordered write from the external memory system must be received at the same time as the mis-speculation is identified in the processor.

Projected Impact:

The erratum causes processor deadlock.

Workarounds:

The recommended workaround is to place a DMB instruction before each Load-Exclusive / Store-Exclusive loop sequence, to ensure that no pending write request can interfere with the

execution of the Load-Exclusive or Store-Exclusive instructions. The implementation of this workaround can be restricted to code regions which have access to Strongly Ordered memory.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. There are some cases where Linux ends up with Strongly Ordered memory (MT_UNCACHED or pgprot_noncached). NXP has checked that these are not used in the BSP. Users should check their application and OS to see if errata conditions met and apply recommended ARM work around if applicable.

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:

A software workaround is possible but it has not been implemented in the Linux BSP yet. BSP functionality may be affected in some configurations and use cases as described above. Users should evaluate their specific use case and apply the recommended workaround to prevent the occurrence of this erratum. Please contact your support channel if you have any questions or concerns.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. There are some cases where Linux ends up with Strongly Ordered memory (MT_UNCACHED or pgprot_noncached). NXP has checked that these are not used in the BSP. Users should check their application and OS to see if errata conditions met and apply recommended ARM work around if applicable.

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 implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The Linux BSP does not support this optional profiling and debug feature. Users may add support for this profiling feature as required, but should ensure the multiple errata impacting the ARM PMU (Performance Monitoring Unit) are considered especially for multi-core usage.

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 which these 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 not implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The BSP has the MMU enabled when it performs BTAC flush in LPM entry. When the kernel is running, the MMU is kept enabled until DSM is entered and the ARM core power is gated.

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 not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. Users should confirm if the conditions apply in their specific OS and apply the ARM recommended workaround if necessary.

ERR009604 ARM (CA9): 845369—Under very rare timing circumstances, transition into streaming mode might create a data corruption

Description:

Under very rare timing circumstances, a data corruption might occur on a dirty cache line that is evicted from the L1 Data Cache due to another cache line being entirely written.

The erratum requires the following conditions:

- The CPU contains a dirty line in its data cache.
- The CPU performs at least four full cache line writes, one of which is causing the eviction of the dirty line.
- Another CPU, or the ACP, is performing a read or write operation on the dirty line.

The defect requires very rare timing conditions to reach the point of failure. These timing conditions depend on the CPU micro-architecture, and are not controllable in software:

- The CPU must be in a transitional mode that might be triggered by the detection of the first two full cache line writes.
- The evicted line must remain stalled in the eviction buffer, which is likely to be caused by a congested write traffic.
- The other coherent agent, either another CPU in the cluster or the ACP, must perform its coherency request on the evicted line while it is in the eviction buffer.

This erratum only occurs when two or more processors are enabled.

Projected Impact:

The erratum might lead to data corruption.

Workarounds:

This erratum can be worked round by setting bit[22] of the undocumented Diagnostic Control Register to 1. This register is encoded as 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,#0x00400000
MCR p15,0,rt,c15,c0,1
```

When this bit is set, the processor is unable to switch into Read-Allocate (streaming) mode, which means this erratum cannot occur.

Setting this bit could possibly result in a visible drop in performance for routines that perform intensive memory accesses, such as memset() or memcpy(). However, the workaround is not expected to create any significant performance degradation in most standard applications.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in Linux BSP codebase starting in release L3.14.38_6qp_ga.

i.MX 6Solo has a single core hence is not impacted by this erratum and a software workaround is not required when in this single core configuration.

ERR009742 ARM: 795769—“Write Context ID" event is updated on read access

Description:

When selected, the Write Context ID event (event 0x0B) of the Performance Monitoring Unit (PMU) increments a counter whenever an instruction that writes to the Context ID register, CONTEXTIDR, is architecturally executed. However this erratum means that an instruction that reads the Context ID register also updates this counter.
The erratum can happen under the following conditions:
1. A PMU counter is enabled, by setting the PMCNTENSET.Px bit to 1 (x identifies a single event counter, and takes a value from 0 to 7).
2. The “Write Context ID” event is mapped to this selected PMU counter:
a. The chosen PMU counter is selected, by setting PMSELR.SEL to x (the same value as in condition 1).
b. The “Write Context ID” event is mapped to this selected PMU, by setting PMXEVTYPER.evtCount to 0x0B.
3. The PMU is enabled, by setting the PMCR.E bit to 1.
4. A read access occurs to the CONTEXTIDR.
In this situation the PMU updates the counter when it should not.

Projected Impact:

The erratum affects the accuracy of the “Write Context ID" event, and its associated PMUEVENT[12] output signal.

Workarounds:

There is no workaround for this erratum. The Linux BSP does not enable this optional profiling feature by default. Users may add support for this profiling feature as required, but should ensure the multiple ARM errata impacting the ARM PMU are considered.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround is not implemented because this erratum will never be encountered in normal device operation.The Linux BSP does not support this optional profiling feature.

ERR009743 ARM: 799770—DBGPRSR Sticky Reset status bit is set to 1 by the CPU debug reset instead of by the CPU non-debug reset

Description:

DBGPRSR.SR, bit [3], is the Sticky Reset status bit. The ARM architecture specifies that the processor sets this bit to 1 when the non-debug logic of the processor is in reset state.
Because of this erratum, the Cortex-A9 processor sets this bit to 1 when the debug logic of the processor is in reset state, instead of when the non-debug logic of the processor is in reset state.

Projected Impact:

- DBGPRSR.SR might not be set to 1 when it should, when the non-debug logic of the processor is in reset state.
- DBGPRSR.SR might be set to 1 when it should not, when the debug logic of the processor is in reset state.

In both cases, the DBGPRSR.SR bit value might be corrupted, which might prevent the debug logic from correctly detecting when the non-debug logic of the processor has been reset.

Workarounds:

No software workaround available as this erratum is related to a debug feature. Users should not rely on the DBGPRSR.SR bit during the debug session.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround is not implemented because this erratum will never be encountered in normal device operation, as this erratum is related to a debug feature.

ERR009858 ARM/PL310: 796171—When data banking is implemented, data parity errors can be incorrectly generated

Description:

When parity is implemented and enabled in the PL310 Level-2 Cache Controller, for each read from the Data RAM, parity of the read data DATARD[255:0] is compared with stored parity bits in dedicated RAMs present on DATAPRD[31:0]. If the comparison does not match, the error is reported using an interrupt mechanism consisting of dedicated registers (Raw and Masked Interrupt registers).

This erratum occurs when the following conditions exist:

1) Parity is enabled (bit[21] of the Auxiliary Control Register is set to 1)

2) Read access latency on Data RAM is programmed with a value > 0x0 (bits [6:4] of the Data RAM Latency

Register)

When the conditions above are met, parity checking between DATARD and DATAPRD occurs during a two cycle window, including one cycle earlier than expected. If, in the early cycle, DATARD and DATAPRD are not stable yet, parity comparison might fail. In this case, an error is reported by the Interrupt registers, where no actual error exists.

Projected Impact:

Because of this erratum, false data parity errors might be reported by the PL310 Level-2 Cache Controller and can cause system instability.

Workarounds:

The following software workarounds can be used to avoid this erratum:

1) Disable parity by setting bit [21] of the Auxiliary Control Register to 0 (this is the default condition).

2) Program the read access latency of the Data RAM to the minimum value acceptable for the implementation plus one (bits [6:4] of the Data RAM Latency Control Register). Note that this workaround can affect performance.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.The Linux BSP does not enable this Parity feature and is disabled by default in all BSP releases. The BSP also ignores any assertion on the PARITYFAIL [7:6] bits by masking the ARM-GIC parity interrupt 125. Please note that the i.MX6 does not support the parity feature (disabled by default) and hence should not be enabled by users.

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:

Software workaround implemented in Linux BSP codebase starting in releases v3.10.9 and v3.0.35.

ERR009219 CCM: Asynchronous clock switching can cause unpredictable behavior

Description:

Certain applications require the source clock of the LVDS Display Bridge (LDB) to be modified to accommodate various display clock frequency requirements. The clock source can be modified by programming an asynchronous clock multiplexer (CCM_CS2CDR[LDB_DIx_CLK_SEL]) in software.

Asynchronous multiplexers or glitchy multiplexers, enable the clock to switch immediately after the multiplexer select is changed. Because both clock sources to the multiplexer are asynchronous, switching the clocks from lone source to the other can cause a glitch to be generated, regardless of the input clock source. This immediate switch of two asynchronous clock domains can cause the output clock to glitch. If the input and output clocks are not gated, this clock glitch can propagate to the logic that follows the clock multiplexer, causing the logic to behave unpredictably.

A clock gate has not been implemented after the asynchronous clock multiplexer for the LDB_DI0_IPU clock and LDB_DI1_IPU clocks. Due to the absence of this clock gate on this LDB_DIx_IPU clock path, a glitch generated when the clock source is switched, can lock up the LDB divider causing a loss of the LDB_DIx_IPU clock under certain conditions.

Projected Impact:

Switching LDB clock sources on an asynchronous clock multiplexer without gating the input and output clock can cause clock glitches to propagate to the logic that follows the clock multiplexers, causing the logic to behave unpredictably. With an ungated input clock, under certain conditions the clock divider in the LDB_DIx_IPU clock path can incorrectly lock up. This can therefore cause a loss of the LDB_DIx_IPU clock which can result in a blank LVDS display screen in the user application.

Workarounds:

The input and output clocks to the asynchronous clock multiplexer are required to be gated prior to switching the source clock. The recommended software workaround is to shut down the clocks to the asynchronous clock multiplexor (CS2CDR: LDB_DIx_CLK_SEL) by disabling the respective PLLs and PFDs prior to performing the clock switch. After the clock switch is performed the input and output clocks of the multiplexer are re-enabled. Users must ensure that the PFDs are reset after the respective PLLs are locked. It is recommended to perform the LDB clock switch early in the boot process to minimize the clocking impact.

Refer to Engineering Bulletin EB821: LDB Clock Switch Procedure and i.MX6 Asynchronous Clock Switching Guidelines for details on the issue and recommended software workaround.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in Linux BSP codebase starting in release L3.0.35_4.1.0.

ERR011100 DCIC: The External Controller Mismatch Indication Signal Outputs Are Not Usable

Description:

The external controller mismatch indication signal is a digital signal intended to switch at the main clock (hsp clock) divided by 4 or divided by 16. Both external controller mismatch indication signal outputs of the DCIC were routed through IOMUX pads that do not support high frequency operation. Because of this, all the possible ALT mode outputs for DCIC1_OUT and DCIC2_OUT cannot switch at the required frequencies making it impossible for an external device to detect match vs. mismatch conditions in any Region of Interest (ROI).

Workarounds:

No hardware workaround possible. Application code should capture and handle the interrupt generated by a mismatch condition. If an indication is needed off-chip, the interrupt handler can signal via a GPIO pin.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

This issue is not applicable to the BSP. Functionality of DCIC output is not used.

ERR009165 eCSPI: TXFIFO empty flag glitch can cause the current FIFO transfer to be sent twice

Description:

When using DMA to transfer data to the TXFIFO, if the data is written to the TXFIFO during an active eCSPI data exchange, this can cause a glitch in the TXFIFO empty signal, resulting in the TXFIFO read pointer (TXCNT) not updating correctly, which in turn results in the current transfer getting resent a second time.

Projected Impact:

Incorrect data transfer when using DMA to transfer data to the eCSPI TXFIFO.

Workarounds:

This errata is only seen when the SMC (Start Mode Control) bit is set. A modified SDMA script with TX_THRESHOLD = 0 and using only the XCH (SPI Exchange) bit to initiate transfers prevents this errata from occurring. There is an associated performance impact with this workaround. Testing transfers to a SPI-NOR flash showed approximately a 5% drop in write data rates and a 25% drop in read data rates.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround implemented in Linux BSP codebase starting in release L3.14.38_6qp_ga.

ERR009535 eCSPI: Burst completion by SS signal in slave mode is not functional

Description:

According to the eCSPI specifications, when eCSPI is set to operate in the Slave mode (CHANNEL_MODE[x] = 0), the SS_CTL[x] bit controls the behavior of burst completion.

In the Slave mode, the SS_CTL bit should control the behavior of SPI burst completion as follows:

- 0—SPI burst completed when (BURST_LENGTH + 1) bits are received
- 1—SPI burst completed when the SS input is negated

Also, in BURST_LENGTH definition, it is stated “In the Slave mode, this field takes effect in SPI transfer only when SS_CTL is cleared.”

However, the mode SS_CTL[x] = 1 is not functional in Slave mode. Currently, BURST_LENGTH always defines the burst length.

According to the SPI protocol, negation of SSB always causes completion of the burst. However, due to the above issue, the data is not sampled correctly in RxFIFO when {BURST_LENGTH+1}mod32 is not equal to {actual burst length}mod32.

Therefore, setting the BURST_LENGTH parameter to a value greater than the actual burst does not resolve the issue.

Projected Impact:

Slave mode with unspecified burst length cannot be supported due to this issue. The burst length should always be specified with the BURST_LENGTH parameter and the SS_CTL[x] should be set to zero.

Workarounds:

There is no workaround except for not using the SS_CTL[x] = 1 option in the Slave mode. The accurate burst length should always be specified using the BURST_LENGTH parameter.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.

ERR009606 eCSPI: In master mode, burst lengths of 32n+1 will transmit incorrect data

Description:

When the ECSPI is configured in master mode and the burst length is configured to a value 32n+1 (where n=0,1, 2,…), the ECSPI will transmit the portions of the first word in the FIFO twice.
For example, if the transmit FIFO is loaded with:
[0] 0x00000001
[1] 0xAAAAAAAA
And the burst length is configured for 33 bits (ECSPIx_CONREG[BURST_LENGTH]=0x020), the ECSPI will transmit the first bit of word [0] followed by the entire word [0], then transmit the data as expected.
The transmitted sequence in this example will be:
[0] 0x00000001
[1] 0x00000001
[2] 0x00000000
[3] 0xAAAAAAAA

Projected Impact:

Incorrect data transmission.

Workarounds:

Do not use burst lengths of 32n+1 (where n=0,1, 2,…).

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The driver limits the burst length up to 32 bits.

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 Linux BSP codebase starting in release 3.0.35_4.0.0 GA.

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 NXP.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in Linux BSP codebase starting in release 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 implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.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:

Software workaround cannot be implemented to mask or workaround this SoC issue. This erratum will result in impacted or reduced functionality as described above.

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 and USB_HSIC to “0,” that is, CMOS input type.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.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:

Software workaround not implemented in the BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. 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 Linux BSP codebase starting in release L3.0.35_4.1.0.

ERR009596 MMDC: ARCR_GUARD bits of MMDC Core AXI Re-ordering Control register (MMDC_MAARCR) doesn't behave as expected

Description:

The ARCR_GUARD bits of MMDC Core AXI Re-ordering Control register (MMDC_MAARCR) are used to ensure better DDR utilization while preventing starvation of lower priority transactions. After reordering is performed on previous read/write DDR transactions, the specific outstanding transaction will first obtain the maximum score in “dynamic score mode" and then wait for additional ARCR_GUARD count before achieving the highest priority. Due to a design issue, the ARCR_GUARD counter doesn't count up to the pre-defined value in the ARCR_GUARD bit field as expected. Therefore, the aging scheme optimizes the transaction reordering only up to the default aging level (15) and assigns a highest priority tag to the outstanding transaction.

Projected Impact:

The aging scheme optimizes the transaction reordering only up to the default aging level (15). No functional issues have been observed with an incorrect setting.

Workarounds:

Software should always program the ARCR_GUARD bits as 4'b0000. That means the accesses which have gained the maximum dynamic score will always become the highest priority after achieving the default highest aging level (15).

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.The Linux BSP releases leave the ARCR_GUARD bits at the default value of 4'b0000.

ERR010481 MMDC: ZQ calibration issue when interfacing to LPDDR2/LPDDR3 memory with two chip selects

Description:

This issue is relevant to processors using the MMDC DDR controller, when attempting to connect to LPDDR2/3 memories that are either single channel (x32)/dual die, or two channel (x64)/quad die. When using these memory devices, the drive strengths of the READ DQS and DQ pins coming from the LPDDR2/3 devices becomes degraded after a short period of time, resulting in corrupted READ operations. The degraded drive strengths that result from the MMDC issuing the ZQ calibration commands to both Chip Selects (CS) of the LPDDR2/3 occur nearly simultaneously, which causes a shared ZQ calibration resistor to give incorrect results.

• This issue can cause data read by the MMDC DDR controller from the DRAM memory to be corrupted due to the incorrect drive strengths.

• This issue only impacts certain LPDDR2/3 memory configurations using multiple chip selects and one ZQ resistor. It does not impact devices with a single CS.

• This issue can impact both single channel and dual channel devices if they have two chip selects (ranks).

• This issue does not impact DDR3/DDR3L memories.

Memory vendors connect the ZQ calibration pin for two dies internally to their parts (for higher densities) forcing the two memory dies to share a single calibration resistor. This is allowed by JEDEC standards, with a caveat that the DDR controller never attempts to issue ZQ command requests to the two memory die at the same time for ZQ calibration. The affected MMDC does the calibration in parallel (at the same time) which causes the calibration to be incorrect. The affected MMDC does not support a mode to run the calibrations serially (one after the other).

Please note that having two ZQ resistors connected on the memory is not sufficient because the memory vendor can have them configured such that both CS/multiple dies can still access the same ZQ resistor at the same time (resulting in the degraded drive strength).

Workarounds:

1) Connect the ZQ calibration pin to a fixed supply:

The JEDEC standards for LPDDR2 give the option to remove the ZQ calibration resistor and simply connect the ZQ calibration resistor pin(s) to VDDCA. The JEDEC specification also specifies the allowed drive strengths of the LPDDR2 device must be achieved over the entire operating temperature range. NXP recommends programming the LPDDR2 memory device to the maximum drive strength in conjunction with the connection of the ZQ calibration resistor pin(s) to the VDDCA on the customer board.

2) Use ZQ SW Calibration:

Users can disable the default automatic ZQ calibrations, both long (ZQCL) and short (ZQCS) calibration, and implement a software patch that triggers ZQCL and ZQCS commands to be sent at staggered times to either chip select. This requires the software to momentarily block memory access to the DRAM before issuing a ZQ calibration command. After the command is issued, the

MMDC will prevent data from being passed to/from the LPDDR2/3 until the required time interval has passed.

3) Use Single CS (RANK) devices:

To completely avoid this issue, customers can use LPDDR2 devices that have a single CS, or are compatible with the MMDC. To achieve higher densities customers can use multiple single-CS (RANK) LPDDR2 devices, however this requires additional board space and routing.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround option of manually performing ZQ calibration to each CS will be integrated in a future Linux BSP codebase.

ERR011421 MMDC: DDR I/O glitches on power-up

Description:

During power-up, glitches have been observed on SDCKE and other DDR I/O signals. Glitches on critical DDR I/O may cause issues during DDR initialization. SDCKE specifically must remain low during power-up per the LPDDR2/3 JEDEC specifications. A glitch on SDCKE during power-up can incorrectly move LPDDR2 into a non-idle state and can impact issuing MRW commends to the LPDDR2 memory.

This issue occurs when the DDR I/O power supply (NVDD_DRAM) is powered up before the internal logic supply to the DDR I/Os (VDD_SOC_CAP).

This issue does not impact DDR3.

Workarounds:

Either the hardware workaround or the software workaround below can be used.

Hardware workaround:

Ensure that VDD_SOC_IN is powered on and VDD_SOC_CAP is stable before powering on NVDD_DRAM.

Software Workaround:

During the DDR initialization script, issue a PRECHARGE ALL command prior to issuing any MRW commands. The PRECHARGE ALL command must be issued to both chip selects if two chip selects are used.

An example of the PRECHARGE ALL command in the DDR initialization script is shown below:

```
########################################################################
# Precharge all command per JEDEC:
# The memory controller may optionally issue a Precharge-All command
# prior to the MRW Reset command.
# This is strongly recommended to ensure a robust DRAM initialization
########################################################################
memory set 0x021b001c 32 0x00008010# PRECHARGE ALL command CS0

# If 2 chip selects are used, issue another PRECHARGE ALL to CS1
# memory set 0x021b001c 32 0x00008018 # PRECHARGE ALL command CS1
```

Proposed Solution:

No fix scheduled.

Linux BSP Status:

The DDR Register Programming Aids for all i.MX6 products have been updated to include the software workaround in the initialization script.

ERR050070 MMDC: Hardware Write Leveling Calibration Error bits MMDC_MPWLGCR[WL_HW_ERRn] are incorrectly de-asserted

Description:

During Auto Hardware Write Leveling, the error status bits (MMDC_MPWLGCR[WL_HW_ERRn]) should be set if an error occurs. These bits are set when an error occurs but then are cleared automatically before software can capture the status. Consequently, the error status bits are not a reliable indication whether a hardware write leveling error has occurred.

Workarounds:

If the hardware write leveling was successful, the MMDC PHY Write Leveling HW Error Register (MMDC_MPWLHWERR) will contain non-zero values for each byte lane used. A zero value for an active byte lane indicates that a hardware write leveling error occurred. Software should use MMDC_MPWLHWERR as the error indication instead of MMDC_MPWLGCR.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

This issue is not applicable to the BSP.

ERR009541 PXP:CSC2 does not perform RGB to YCbCr and RGB to YUV conversions

Description:

When performing RGB-to-YUV conversions, CSC2 can only output a result from 0 to 255 (8 bits), which does not meet the YUV range requirements (9 bits required) and the parameters d0/d1/d2 cannot meet the YCbCr requirements (17 bits required).

Projected Impact:

RGB-to-YUV or RGB-to-YCbCr conversions must be performed by a resource other than the PXP.

Workarounds:

None

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workarounds not applicable to the Linux BSP.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.

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:

Software workaround cannot be implemented to mask or workaround this SoC issue. This erratum will result in impacted or reduced functionality as described above.

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.

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 implemented in Linux BSP codebase starting in u-boot v2013.04 and in L3.10.9_1.0.0_alpha release.

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

ERR009678 ROM: SD/EMMC/NAND prematurely times out during boot

Description:

RTC_XTALI picks up noise during crystal start up, during power up, and the boot sequence. Once the RTC crystal is stable and running, no noise interference is observed. The noise causes the RTC oscillator to output noise to an automatic multiplexer where the internal ring oscillator and the RTC oscillator are connected. If the noise contains frequency components higher than approximately 500 kHz, the output of the automatic multiplexer sends high frequency signals which causes General Purpose Timer (GPT) to count at a significantly faster rate than 32 kHz and causing a premature GPT timeout interrupt. The premature interrupt results in the boot ROM being redirected to USB serial download mode.

Projected Impact:

System boot failure can be observed with SD card, eMMC, and NAND primary boot.

Workaround:

1. Connect RTC_XTALI to an external 32 kHz clock source. The external clock source should be stable before POR_B is de-asserted.
2. Remove the 32 kHz crystal and connect RTC_XTALI to GND. This will use internal ring oscillator. This workaround is not recommended for designs requiring an accurate 32 kHz clock.
3. Delay POR_B de-assertion until after the 32 kHz crystal is stable (approximately 300 ms to 500 ms from VDD_SNVS_IN ramp timing; this timing is dependent on the crystal start up timing characteristics).

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround cannot be implemented to mask or workaround this SoC issue. This erratum will result in impacted or reduced functionality as described above.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.

ERR008990 SSI: Channel swap in single FIFO mode when an underrun or overrun occurs

Description:

In I2S mode, with one FIFO in use, data is in the format left channel, right channel, left channel, right channel. If an under-run occurs, then the left and right channels will transmit the same previous data until new data is written to the FIFO. If the new data is valid as the right channel starts to transmit, a channel swap occurs. Likewise when receiving data, if an overrun occurs and the FIFO is emptied as the right channel data is being received, a channel swap occurs.

Projected Impact:

When using SSI in I2S mode, operation is limited to two-channel mode.

Workarounds:

Use SSI in two-channel mode (TCH_EN = 1) with two FIFOs enabled (TFEN1=1, TFEN0=1). With two FIFOs in use, left channel transmit data is from FIFO0, right channel transmit data is from FIFO1. When an under-run occurs, the left channel will transmit the previous data in FIFO0, while the right channel will transmit the previous data in FIFO1. When new data is written into FIFO0/FIFO1, the left channel data is always from FIFO0, and right channel data is always from FIFO1, preventing channel swapping from occurring. Likewise when receiving data, left channel data is always stored in FIFO0 and right channel data is always stored in FIFO1.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround implemented in Linux BSP codebase starting in release L3.10.53_1.1.0_ga.

ERR011121 System Boot: EIM NOR boot failed on closed part if image targeted into OCRAM

Description:

A boot image corruption can result in a boot failure for an EIM NOR boot devices under specific conditions as described below. The boot failure only occurs if all conditions below are satisfied.

The following i.MX products are affected:

i.MX 6SoloLite silicon revision 1.4

Conditions:

There are four specific conditions that result in this boot failure.

1) i.MX device with the silicon revision listed above. Earlier silicon revisions are not affected.

2) An EIM NOR boot device is used.

3) The device is a security enabled configuration (SEC_CONFIG[1] eFUSE is programmed).

4) The boot image start address (specified in boot data) targets internal memory on-chip RAM (OCRAM) space.

This issue can result in EIM NOR boot failure and possible entry into Serial Downloader mode.

Projected Impact:

NOR boot fail.

Workarounds:

Because the boot failure only occurs when the boot image start address targets OCRAM space, the recommended workarounds are for users to ensure the NOR boot image runs from DDR or executes in place (XIP).Proposed Solution:

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround not implemented in BSP. Functionality where the erratum may manifest itself is not used.

ERR004535 USB: USB suspend and resume flow clarifications

Description:

In device mode, The PHY can be put into Low Power Suspend when the device is not running or the host has signaled suspend. The PHY Low power suspend bit (PORTSC1.PHCD) will be cleared automatically when the host initials resume. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend when the downstream device has been placed into suspend mode (PORTSC1.SUSP) or when no downstream device is connected. Low power suspend is completely under the control of software.

To place the PHY into Low power mode, software needs to set PORTSC1.PHCD bit, set all bits in USBPHY_PWD register and set the USBPHY_CTRL.CLKGATE bit.

When a remote wakeup occurs after the Suspend (SUSP) bit is set while the PHY Low power suspend bit (PHCD) is cleared, a USB interrupt (USBSTS.PCI) will be generated. In this case, the PHCD bit will NOT be set because of the interrupt. However, if a remote wakeup occurs after the PHCD bit is set while the USB PHY Power-Down Register (USBPHY_PWD) and the UTMI clock gate (USBPHY_CTRL.CLKGATE) bit is cleared, a remote wakeup interrupt will be generated. In this case, all the bits in the HW_USBPHY_PWD register and the USBPHY_CTRL.CLKGATE bit will be set, even after the remote wakeup interrupt is generated, which is incorrect.

Projected Impact:

Resume error, if the correct flow is not adhered to.

Workarounds:

To place the USB PHY into low power suspend mode, the following sequence should be performed in an atomic operation (interrupts should be disabled during these three steps):

1. Set the PORTSC1.PHCD bit
2. Set all bits in the USBPHY_PWD register
3. Set the USBPHY_CTRL.CLKGATE bit

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in Linux BSP codebase starting in release L3.0.35_4.1.0.

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 implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used.

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 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

ERR007881 USB: Timeout error in Device mode

Description:

If a receive FIFO overrun occurs (due to a busy condition on the system bus) when the USB controller is in Device mode, the controller may stop responding to host tokens, causing current transactions to time out. This situation will be recovered after FIFO is not overrun.

Projected Impact:

Implementing the workaround shown will prevent receive FIFO overruns, but cause a 10%-30% impact to USB performance.

Workarounds:

Set Stream Disable mode (USB_nUSBMODE[SDIS]=1) to prevent receive FIFO overruns.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in Linux BSP codebase starting in L3.10.53_1.1.0_ga.

ERR010822 USB: USB host may not respond to RX data

Description:

Under rare conditions, the USB controller operating in host mode may fail to respond to an incoming packet from a device when two or more hubs are between the USB host and device. This configuration causes extra bits in the data stream after the EOP. The issue does not occur without a hub connection because there are no additional bits after EOP.

The issue occurs under the following conditions:

1. The TX clock from the device is faster than the RX clock at the host, and
2. Extra bits are present in the received data stream after EOP.

When the issue occurs, the USB controller will not respond to the current RX packet (neither ACK nor NAK) and the packet is discarded. For a non-ISO transfer, the TX side should resend the packet (per the USB specification requirements). For an ISO transfer, the packet will be lost.

Condition 1) above can cause occurrences where the USB controller will see two bit transitions within one 480 MHz cycle. The USB specification requires clock accuracy of 500 ppm, so the maximum difference between the TX and RX sides is 1000 ppm, or 1/1000 bits. The issue is rare since two bit transitions within one clock cycle must also coincide with the EOP. The occurrence of the issue is further reduced with shorter packet lengths, but there is no known maximum packet length that avoids the issue entirely.

When this issue happens, a UTMI_RXERROR is generated.

Workarounds:

For a non-ISO transfer, the device will retransmit the packet per the USB specification requirements so the system can handle this case automatically.

For a non-ISO transfer, the host will discard the packet and the data is lost. There is no known software workaround for this case.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround implemented in the Linux BSP.

ERR004536 uSDHC: ADMA Length Mismatch Error may occur for longer read latencies

Description:

When a multi-block read command is triggered, the controller starts to send read commands and receives data from the card. After one block of data is received, the expected block count number will be updated (minus 1). Meanwhile the DMA engine starts to fetch the ADMA descriptors through the AHB bus. The descriptor contains two AHB SINGLE bus accesses for ADMA2 and one AHB SINGLE bus access for ADMA1. The DMA engine then loads the expected block count. If the total latency of these AHB SINGLE bus accesses is longer than the latency for one block to be read from the card, an incorrect block count is loaded by the DMA engine.

Projected Impact:

If the total latency of these AHB SINGLE bus accesses is longer than the latency of one block being read from the card in ADMA mode, the incorrect block count will be loaded by the DMA engine.

Workarounds:

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:

A software workaround is possible but it hasn't been implemented in the Linux BSP yet. BSP functionality may be affected in some configurations and use cases as described above. Users should evaluate their specific use case and apply the recommended workaround to prevent the occurrence of this erratum. Please contact your support channel if you have any questions or concerns.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number: IMX6SLCE
Rev. 5
02/2019