Freescale Semiconductor Inc.
Chip Errata

IMX6SXCE
Rev. 1, 04/2016

Chip Errata for the i.MX 6SoloX

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

Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Changes
0	02/2015	Initial public release
1	02/2016	• **Added errata:** • ERR003741 • ERR004324 • ERR005200 • ERR005829 • ERR007006 • ERR007881 • ERR008506 • ERR009535 • ERR009541 • ERR009572 • ERR009596 • ERR009606 • ERR009636 • ERR009742 • ERR009743 • ERR009858 **Changed errata: Section revised:** • ERR003717: Linux BSP Status • ERR003718: Linux BSP Status • ERR003719: Linux BSP Status • ERR003721: Linux BSP Status • ERR003723: Linux BSP Status • ERR003724: Description and Linux BSP Status • ERR003725: Linux BSP Status • ERR003726: Linux BSP Status • ERR003727: Linux BSP Status • ERR003728: Linux BSP Status • ERR003729: Linux BSP Status • ERR003730: Linux BSP Status • ERR003731: Linux BSP Status • ERR003732: Linux BSP Status • ERR003733: Linux BSP Status • ERR003734: Linux BSP Status • ERR003735: Linux BSP Status • ERR003736: Linux BSP Status • ERR003737: Linux BSP Status • ERR003738: Linux BSP Status • ERR003739: Linux BSP Status • ERR004326: Linux BSP Status • ERR004327: Linux BSP Status • ERR005175: Linux BSP Status • ERR005183: Linux BSP Status • ERR005185: Linux BSP Status • ERR005187: Linux BSP Status • ERR005198: Linux BSP Status • ERR005382: Linux BSP Status • ERR005385: Updated wording in all sections • ERR005386: Linux BSP Status • ERR005387: Linux BSP Status • ERR007007: Linux BSP Status • ERR007008: Linux BSP Status

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

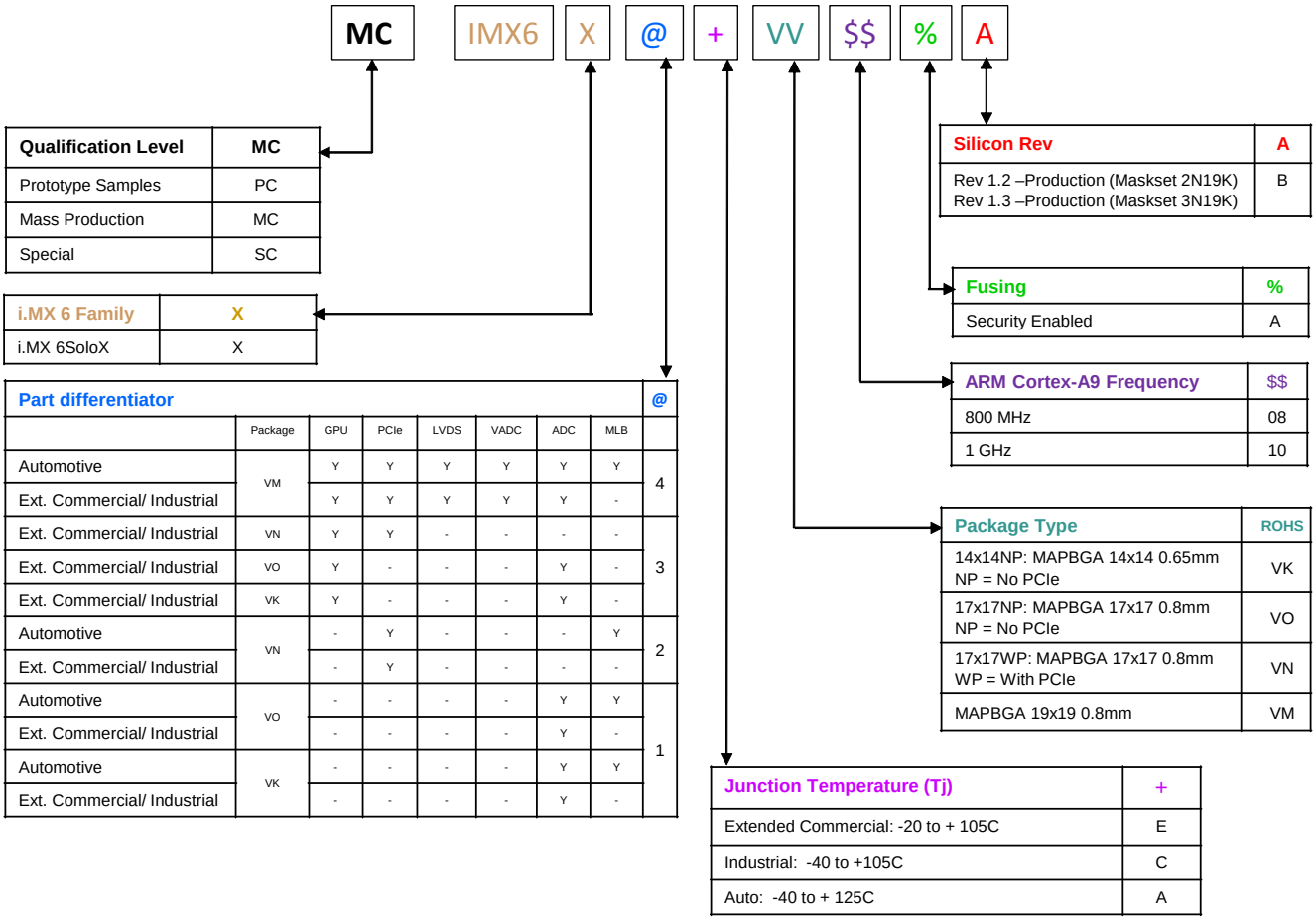

Figure 1. Revision Level to Part Marking Cross-Reference

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

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

Table 2. Summary of Silicon Errata

Errata	Name	Solution	Page
ADC			
ERR008989	ADC: ADC may hang in hardware trigger mode with overwrite disabled if the COCO flag is not cleared before finish of next conversion.	No fix scheduled	10
ARM® – Cortex A9			
ERR003717	ARM: 740657—Global Timer can send two interrupts for the same event	No fix scheduled	11
ERR003718	ARM: 743622—Faulty logic in the Store Buffer may lead to data corruption	No fix scheduled	13
ERR003719	ARM/MP: 751469 — Overflow in PMU counters may not be detected	No fix scheduled	15
ERR003721	ARM: 751473—Under very rare circumstances, Automatic Data prefetcher can lead to deadlock or data corruption	No fix scheduled	16
ERR003723	ARM: 751476—May miss a watchpoint on the second part of an unaligned access that crosses a page boundary	No fix scheduled	17
ERR003724	ARM: 754322—Possible faulty MMU translations following an ASID switch	No fix scheduled	18
ERR003725	ARM: 725631—ISB is counted in Performance Monitor events 0x0C and 0x0D	No fix scheduled	20
ERR003726	ARM: 729817—MainID register alias addresses are not mapped on Debug APB interface	No fix scheduled	21
ERR003727	ARM: 729818—In debug state, next instruction is stalled when sdabort flag is set, instead of being discarded	No fix scheduled	22
ERR003728	ARM: 740661—Event 0x74 / PMUEVENT[38:37] may be inaccurate	No fix scheduled	23
ERR003729	ARM: 740663—Event 0x68 / PMUEVENT[9:8] may be inaccurate	No fix scheduled	24
ERR003730	ARM: 743623—Bad interaction between a minimum of seven PLDs and one Non-Cacheable LDM can lead to a deadlock	No fix scheduled	26
ERR003731	ARM: 743626—An imprecise external abort, received while the processor enters WFI, may cause a processor deadlock	No fix scheduled	28
ERR003732	ARM: 751471—DBGPCSR format is incorrect	No fix scheduled	29
ERR003733	ARM: 751480—Conditional failed LDREXcc can set the exclusive monitor	No fix scheduled	31
ERR003734	ARM: 752519—An imprecise abort may be reported twice on non-cacheable reads	No fix scheduled	32
ERR003735	ARM: 754323—Repeated Store in the same cache line may delay the visibility of the Store	No fix scheduled	33
ERR003736	ARM: 756421—Sticky Pipeline Advance bit cannot be cleared from debug APB accesses	No fix scheduled	35
ERR003737	ARM: 757119—Some “Unallocated memory hint” instructions generate an UNDEFINED exception instead of being treated as NOP	No fix scheduled	36

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ERR003738	ARM/MP: 751475—Parity error may not be reported on full cache line access (eviction / coherent data transfer / cp15 clean operations)	No fix scheduled	37
ERR003739	ARM: 751470—Imprecise abort on the last data of a cache linefill may not be detected	No fix scheduled	38
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	39
ERR004324	ARM/MP: 761319—Ordering of read accesses to the same memory location may not be ensured	No fix scheduled	40
ERR004326	ARM/MP: 761321—MRC and MCR are not counted in event 0x68	No fix scheduled	41
ERR004327	ARM/MP: 764319—Read accesses to DBGPRSR and DBGOSLSR may generate an unexpected UNDEF	No fix scheduled	42
ERR005175	ARM/MP: 771221—PLD instructions may allocate data in the Data Cache regardless of the Cache Enable bit value	No fix scheduled	43
ERR005183	ARM/MP: 771224—Visibility of Debug Enable access rights to enable/disable tracing is not ensured by an ISB	No fix scheduled	44
ERR005185	ARM/MP: 771225—Speculative cacheable reads to aborting memory region clear the internal exclusive monitor, may lead to livelock	No fix scheduled	45
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	47
ERR005198	ARM/PL310: 780370—DATAERR, TAGERR, and Tag parity errors are incorrectly sampled by the eviction buffer, leading to data corruption	No fix scheduled	48
ERR005200	ARM/MP: 765569—Prefetcher can cross 4 KB boundary if offset is programmed with value 23	No fix scheduled	50
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	51
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	52
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	53
ERR005386	ARM/MP: 782773—Updating a translation entry to move a page mapping might erroneously cause an unexpected translation fault	No fix scheduled	55
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	57
ERR005391	ARM: Debug CTI interrupt can cause a system deadlock when power gating the core	No fix scheduled	58
ERR007006	ARM/MP:794072-- Short loop including a DMB instruction might cause a denial of service	No fix scheduled	59
ERR007007	ARM/MP: 794073—Speculative instruction fetches with MMU disabled might not comply with architectural requirements	No fix scheduled	61

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
ERR007008	ARM/MP: 794074—A write request to Uncacheable Shareable memory region might be executed twice	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
ERR009858	ARM/PL310: 796171 When data banking is implemented, data parity errors can be incorrectly generated	No fix scheduled	66
ARM® – Cortex M4			
ERR006940	Cortex M4: 776924—VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	No fix scheduled	67
ERR007581	ARM(CM4): The write data to TCRAM through the backdoor may be corrupted if the BUSY state is inserted during the transfer	No fix scheduled	69
CCM			
ERR007265	CCM: When improper low-power sequence is used, the SoC enters low power mode before the ARM core executes WFI	No fix scheduled	70
ERR006223	CCM: Failure to resume from Wait/Stop mode with power gating	No fix scheduled	71
DEXSC MRVS			
ERR007782	DEXSC MRVS (Memory Region Violation Register): erroneously logs the write address even though the domain access violation is caused by a simultaneous read access	No fix scheduled	72
eCSPI			
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
ENET			
ERR007885	ENET: Potential sequencing issue with TDAR in Multi-Queue	No fix scheduled	75
ESAI			
ERR008000	ESAI: ESAI may encounter channel swap when overrun/underrun occurs	No fix scheduled	76
FlexCAN			
ERR005829	FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process	No fix scheduled	77
GPMI			
ERR008650	GPMI: BCH ECC Randomizer will fail when reading an erased page.	No fix scheduled	79

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
I2C			
ERR007805	I2C: When the I2C clock speed is configured for 400 kHz, the SCL low period violates the I2C specification	No fix scheduled	80
MMDC			
ERR009596	MMDC: ARCR_GUARD bits of MMDC Core AXI Re-ordering Control register (MMDC_MAARCR) doesn't behave as expected	No fix scheduled	81
ERR009636	MMDC: Random data corruption during reads from DDR memory	Fix in silicon revision 1.3	82
PCIe			
ERR003747	PCIe: Reading the Segmented Buffer Depth Port Logic registers returns all zeros	No fix scheduled	83
ERR005184	PCIe: Clock pointers can lose sync during clock rate changes	No fix scheduled	84
ERR007520	PCIe: L1/L2 entry negotiation not restarted after interruption	No fix scheduled	86
ERR007554	PCIe: MSI Mask Register Reserved Bits not read-only	No fix scheduled	87
ERR007555	PCIe: iATU—Optional programmable CFG Shift feature for ECAM is not correctly updating address	No fix scheduled	88
ERR007556	PCIe: Core Delays Transition From L0 To Recovery After Receiving Two TS OS And Erroneous Data	No fix scheduled	89
ERR007557	PCIe: Extra FTS sent when Extended Synch bit is set	No fix scheduled	90
ERR007559	PCIe: Core sends TS1 with non-PAD lane number too early in Configuration.Linkwidth.Accept State	No fix scheduled	91
ERR007573	PCIe: Link and lane number-match not checked in recovery	No fix scheduled	92
ERR007574	PCIe: Loopback—Core does not clear available credits when link goes down	No fix scheduled	93
ERR007575	PCIe: LTSSM delay when moving from L0 to recovery upon receipt of insufficient TS1 Ordered Sets	No fix scheduled	94
ERR007577	PCIe: DLLP/TLP can be missed on RX path when immediately followed by EIOS	No fix scheduled	95
PXP			
ERR009541	PXP: CSC2 does not perform RGB to YCbCr and RGB to YUV conversions	No fix scheduled	96
QuadSPI			
ERR007785	QuadSPI: Incorrect data returned in multiple flash use case when the read crosses flash boundary	No fix scheduled	97
ERR008611	QuadSPI: When an abort occurs to a suspended low-priority flash transaction during a new AHB request from the same low-priority master of a different address, then the new flash transaction starts from a wrong address.	No fix scheduled	98

Table 2. Summary of Silicon Errata (continued)

Errata	Name	Solution	Page
RDC			
ERR007637	RDC: Clearing violating address bit in MRVS (memory region violation status) register can only be from domain 0	No fix scheduled	99
ERR009572	RDC: Access to RDC registers will cause the CPU to hang if the PCIE_DISABLE fuse has disabled PCIe	No fix scheduled	100
ROM			
ERR008506	ROM: Incorrect NAND BAD Block Management	No fix scheduled	101
USB			
ERR007881	USB: Timeout error in Device mode	No fix scheduled	102
ERR009059	USB: USB_OTG1_PWR (GPIO1_IO09) will output high during reset	No fix scheduled	103

Table 3 lists the ARM core errata that are excluded from this errata document.

Table 3. Excluded ARM Errata

ARM Errata Reference	Title	Reason why excluded from this errata document
743625	A coherent ACP request might interfere with a non-cacheable SWP/SWPB from the processor, Potentially causing deadlock	i.MX6 does not support the ACP (Accelerator Coherency Port) hence errata not applicable.
754319	A sequence of cancelled Advanced-SIMD or VFP stores might deadlock	Errata was fixed in the Floating Point Unit (FPU) Revision 0x4 which is used across i.MX6, hence not applicable.
754320	A cancelled Advanced-SIMD or VFP load multiple of more than 8 beats might deadlock	Errata was fixed in the Floating Point Unit (FPU) Revision 0x4 which is used across i.MX6, hence not applicable.
745320	A Floating Point write following a failed conditional read might write corrupted data	Errata was fixed in the Floating Point Unit (FPU) Revision 0x4 which is used across i.MX6, hence not applicable.
751469	Overflow in PMU counters may not be detected	i.MX6 does not support PMU (Performance Monitoring Unit) hence this ARM errata is not applicable.
751475	Parity error may not be reported on full cache line access (eviction / coherent data transfer / cp15 clean operations)	i.MX6 does not implement parity hence errata not applicable. The parity feature is disabled by default and should not be enabled.
761321	MRC and MCR are not counted in event 0x68	i.MX6 does not support PMU (Performance Monitoring Unit) hence this ARM errata is not applicable.
764269	Under very rare circumstances, a sequence of at least three writes merging in the same 64-bit address range might cause data corruption	ARM has not managed to reproduce the failure in actual Silicon and no software workaround available for this erratum.

Table 3. Excluded ARM Errata (continued)

791420	Possible denial of service for coherent requests on a cache line which is continuously written by a processor	Freescale cannot disclose this errata per ARM requirements, however software workaround has been implemented in the Freescale Linux BSP for this erratum. OS vendors/users must approach ARM if further information is required.
756420	Instruction Cache parity error reporting on PARITYFAIL[5:4] output is one cycle earlier than the other PARITY FAIL bits	i.MX6 does not implement parity hence errata not applicable. The parity feature is disabled by default and should not be enabled.
732672	An abort on the second part of a double linefill can cause data corruption on the first part	Freescale cannot disclose this errata per ARM requirements, however software workaround has been implemented in the Freescale Linux BSP for impacted devices. OS vendors/users must approach ARM if further information is required.

ERR008989 ADC: ADC may hang in hardware trigger mode with overwrite disabled if the COCO flag is not cleared before finish of next conversion.

Description:

If ADC is working in hardware trigger mode and ADC_CFG[OVWREN] bit is 0, i.e. overwrite of data registers are disabled, then ADC may hang if the COCO flag of the previous conversion is not cleared by reading data or other means prior to finish of next conversion.

Projected Impact:

Hardware trigger mode with overwrite disabled is not available for use.

Workarounds:

Always use ADC_CFG[OVWREN]=1 setting in hardware trigger mode

Proposed Solution:

No fix scheduled

Linux BSP Status:

The BSP does not enable hardware trigger mode.

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 needed 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.

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 imx_3.0.35_4.1.0.

ERR003719 ARM/MP: 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 needed because this erratum will not be encountered in normal device operation. The Freescale 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.

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 needed 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:

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

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 integrated in Linux BSP codebase starting in release imx_3.0.35_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 needed because this erratum will not be encountered in normal device operation.The Freescale 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.

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 needed 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 needed 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 needed because this erratum will not be encountered in normal device operation. The Freescale 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.

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 needed because this erratum will never be encountered in normal device operation.The Freescale 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.

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

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

Description:

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

Conditions:

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

Projected Impact:

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

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

Workarounds:

There is no practical software workaround for the failure.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround can be implemented to mask or workaround this SoC issue. This erratum will result in impacted or reduced functionality as described above.

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 needed 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/MP: 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:

The workaround for this erratum is to implement an abort granularity of (at least) one cache line for all cacheable memory regions which can abort.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround not implemented in the BSP.

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.

ERR004324 ARM/MP: 761319—Ordering of read accesses to the same memory location may not be ensured

Description:

The ARM architecture, and general rules of coherency, requires reads to the same memory location to be observed in order.

Due to some internal replay path mechanisms, the Cortex-A9 can see a read access bypassed by a following read access to the same memory location, thus not observing the values in program order.

Conditions:

It can only happen on memory regions marked as Normal Memory Write-Back Shared.

Projected Impact:

The erratum leads to data coherency failure.

Workarounds:

The vast majority of multi-processing code is not written in a style which exposes the erratum. So, the erratum is expected to affect very specific areas of code which rely on this read ordering behavior.

A first workaround for this erratum consists in using LDREX instead of standard LDR in volatile memory places where a strict read ordering is required.

An alternative solution consists in inserting a DMB between the affected LDR which requires this strict ordering rule. This is the recommended workaround for tool chains integration.

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 ensure their tool chain has the recommended workaround. For more information about integrating the workaround inside tool chains, please refer to the Programmer Advice Notice related to this erratum, ARM UAN 0004A.

(http://infocenter.arm.com/help/topic/com.arm.doc.uan0004a/UAN0004A_a9_read_read.pdf)

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 Freescale 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.

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 needed 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 Freescale 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.

ERR005200 ARM/MP: 765569—Prefetcher can cross 4 KB boundary if offset is programmed with value 23

Description:

When prefetch feature is enabled (bits [29:28] of the Auxiliary or Prefetch Control Register set HIGH), the prefetch offset bits of the Prefetch Control Register (bits [4:0]) permits to configure the advance taken by the prefetcher compared to the current cache line. Refer to the TRM for more information. One requirement for the prefetcher is not to go beyond a 4 KB boundary. If the prefetch offset is set to 23 (5'b10111), this requirement is not fulfilled and the prefetcher can cross a 4 KB boundary.

This problem occurs when the following conditions are met:

1. One of the Prefetch Enable bits (bits [29:28] of the Auxiliary or Prefetch Control Register) is set HIGH.
2. The prefetch offset bits are programmed with value 23 (5'b10111).

Projected Impact:

When the conditions above are met, the prefetcher can issue linefills beyond a 4 KB boundary compared to original transaction. This can cause system issues because those linefills can target a new 4 KB page of memory space, regardless of page attribute settings in L1 MMU.

Workarounds:

A workaround for this erratum is to program the prefetch offset with any value except 23.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround integrated in Linux BSP codebase starting in release imx_3.0.35_4.1.0. BSP software workaround sets prefetch offset to 0 or 15 to avoid this erratum.

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 Freescale 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.

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 integrated in Linux BSP codebase starting in release imx_3.0.35_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 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). Freescale 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:

Workaround possible but not implemented in the BSP, impacting functionality as described above.The Linux community has not incorporated a workaround for this erratum

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). Freescale 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 applicable to the BSP

ERR007006 ARM/MP:794072-- Short loop including a DMB instruction might cause a denial of service

Description:

A processor which continuously executes a short loop containing a DMB instruction might prevent a CP15 operation broadcast by another processor from making further progress, thus causing a denial of service.

The erratum requires the following conditions:

- Two or more processors are working in SMP mode (ACTLR.SMP=1)
- One of the processors continuously executes a short loop containing at least one DMB instruction.
- Another processor executes a CP15 maintenance operation that is broadcast. This requires that this processor has enabled the broadcasting of CP15 operations (ACTLR.FW=1)

For the erratum to occur, the short loop containing the DMB instruction must meet all of the following additional conditions:

- No more than 10 instructions other than the DMB are executed between each DMB
- No nonconditional Load or Store, or conditional Load or Store which pass the condition code check, are executed between each DMB

When all the conditions for the erratum are met, the short loop creates a continuous stream of DMB instructions. This might cause a denial of service, by preventing the processor executing the short loop from executing the received broadcast CP15 operation. As a result, the processor that originally executed the broadcast CP15 operation is stalled until the execution of the loop is interrupted.

Note that because the process issuing the CP15 broadcast operation cannot complete operation, it cannot enter any debug mode, and cannot take any interrupt. If the processor executing the short loop also cannot be interrupted—for example if it has disabled its interrupts—or if no interrupts are routed to this processor, this erratum might cause a system livelock.

Projected Impact:

The erratum might create performance issues, or in the worst case it might cause a system livelock, if the processor executing the DMB is in an infinite loop that cannot be interrupted.

Workarounds:

This erratum can be worked around by setting bit[4] of the undocumented Diagnostic Control register to 1. This register is encoded as CP15 c15 0 c0 1.

This 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,#0x10
MCR p15,0,rt,c15,c0,1
```

When it is set, this bit causes the DMB instruction to be decoded and executed like a DSB.

Using this software workaround is not expected to have any impact on the overall performance of the processor on a typical code base.

Other workarounds are also available for this erratum, to either prevent or interrupt the continuous stream of DMB instructions that causes the deadlock. For example:

- Inserting a nonconditional Load or Store instruction in the loop between each DMB

- Inserting additional instructions in the loop, such as NOPs, to prevent the processor from seeing back-to-back DMB instructions.

- Making the processor executing the short loop take regular interrupts.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround integrated in Linux BSP codebase starting in release imx_3.0.35_4.1.0.

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 needed 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 kernel is running, the MMU is kept enabled until DSM is entered and 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.

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 Freescale 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 Freescale 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 Freescale 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.

ERR006940 Cortex M4: 776924—VDIV or VSQRT instructions might not complete correctly when very short ISRs are used

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

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

Projected Impact:

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

Conditions:

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

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

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

Workarounds:

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

There are two workarounds:

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

Proposed Solution:

No fix scheduled

MQX BSP Status:

Software workaround implemented since Alpha release

ERR007581 ARM(CM4): The write data to TCRAM through the backdoor may be corrupted if the BUSY state is inserted during the transfer

Description:

There is an AHB backdoor interface in the Cortex-M4 platform for all the other masters except CM4 to access TCRAM. TCRAM data may be corrupted through this AHB backdoor writing access.

Projected Impact:

TCRAM data may be corrupted.

Conditions:

TCRAM data may be corrupted when all these conditions are met:

1. A write transaction is issued through the AHB backdoor interface.
2. The "BUSY" transfer type is inserted in the middle of bursts of transfers.
3. The "WDATA" is changed during the BUSY cycles.
4. The burst type is neither "SINGLE" nor "INCR".

Workarounds:

Software set "force_incr" bit to 1 in the "ahb_cntl" register of pl301_wakup slave 6 AMIB. This bit can force the transfer burst type always be "INCR" on the AHB backdoor interface.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented since Alpha release.

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

Description:

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

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

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

Projected Impact:

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

Workarounds:

Software workaround:

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

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

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

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

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:

$RBC\_COUNT \times (1/32K\ RTC\ Frequency) \geq (25 + PDNSCR\_SW2ISO) \times (1/IPG\_CLK\ Frequency)$

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

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Implemented in BSP version Post GA L3.0.35_1.1.0

ERR007782 DEXSC MRVS (Memory Region Violation Register): erroneously logs the write address even though the domain access violation is caused by a simultaneous read access

Description:

When there is a domain access violation for the read address and when “wvalid" is high during that time, then the MRVS register is updated with the write address and the write domain ID instead of the read address and the read domain ID even though the write address is not the violating register address.

When the write address receives a domain access violation then the MRVS register does not incorrectly log the access violation.

Projected Impact:

This is a corner case that causes an inconvenience to the user when debugging a read address with a domain access violation when the MRVS register is used to locate the address that caused the domain violation.

Workarounds:

The user can disable domain access from the master which shows write address domain access violations, then the true domain access violation address and ID for the read access with the violation shows up correctly.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

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.

ERR007885 ENET: Potential sequencing issue with TDAR in Multi-Queue mode

Description:

When the 10/100-Mbps Ethernet Media Access Control (ENET MAC) module is in Multi-queue mode, there is a potential sequencing issue between the module clearing the ENET Transmit Descriptor Active Register (ENET_TDARn_TDAR) bit and the software setting it. This can cause the module to hang.

Projected Impact:

Under certain timing conditions, the ENET module may hang.

Workarounds:

ENET_TDARn_TDAR should be set by software after it is cleared by the ENET. This is achieved by introducing a short delay after a new Transmit Buffer Descriptor (TxBD) is prepared and written into a designated memory.

— Software prepares a new TxBD and stores/writes it into a designated memory

— Software introduces a delay by reading the relevant ENET_TDARn_TDAR 4 times as shown by the following pseudo-code :

For (i=0; i<4; i++) // 4 Reads should be sufficient

```
{
//Read TDAR
If (TDAR == 0)
{
tdar_trigger = 1
exit_for_loop
}
else
tdar_trigger = 0
end
}
```

If (tdar_trigger)

Set TDAR = 1 (i.e., set ENET_TDARn_TDAR to 1)

Else

Do_nothing (i.e., don’t trigger TDAR)

End

Therefore the software can set the TDAR bit as soon as it is detected as zero.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR008000 ESAI: ESAI may encounter channel swap when overrun/underrun occurs

Description:

While using ESAI transmit or receive and an underrun/overrun happens, channel swap may occur. The only recovery mechanism is to reset the ESAI.

Projected Impact:

Underrun/overrun may cause audio channel swap.

Workarounds:

Underrun/overrun in the ESAI should be prevented at the system level. If channel swap occurs, the ESAI must be reset according to the reset procedure documented in the reference manual.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR005829 FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process

Description:

FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process. The following conditions are necessary for the issue to occur:

- Only one message buffer is configured to be transmitted
- The write which enables the message buffer to be transmitted (write on Control/Status word) happens during a specific clock during the arbitration process.
- After this arbitration process occurs, the bus goes to the Idle state and no new message is received on the bus.

For example:

1. Message buffer 13 is deactivated on RxIntermission (write 0x0 to the CODE field from the Control/Status word) [First write to CODE]
2. Reconfigure the ID and data fields
3. Enable the message buffer 13 to be transmitted on BusIdle (write 0xC on CODE field) [Second write to CODE]
4. CAN bus keeps in Idle state
5. No write on the Control/Status from any message buffer happens.

During the second write to CODE (step 3), the write must happen one clock before the current message buffer 13 to be scanned by arbitration process. In this case, it does not detect the new code (0xC) and no new arbitration is scheduled.

The problem can be detected only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is no issue if any of the conditions below holds:

- Any message buffer (either Tx or Rx) is reconfigured (by writing to its CS field) just after the Intermission field.
- There are other configured message buffers to be transmitted
- A new incoming message sent by any external node starts just after the Intermission field.

Projected Impact:

FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment.

Workarounds:

To transmit a CAN frame, the CPU must prepare a message buffer for transmission by executing the following standard 5-step procedure:

1. Check if the respective interrupt bit is set and clear it.
2. If the message buffer is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control/Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the

interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted. If backwards compatibility is desired (MCR[AEN] bit negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the message buffer, but then the pending frame might be transmitted without notification.

3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control and CODE fields of the Control/Status word to activate the message buffer.
6. The workaround consists of executing two extra steps:
7. Reserve the first valid mailbox as an inactive mailbox (CODE=0b1000). If RX FIFO is disabled, this mailbox must be message buffer 0. Otherwise, the first valid mailbox can be found using the "RX FIFO filters" table in the FlexCAN chapter of the chip reference manual.
8. Write twice INACTIVE code (0b1000) into the first valid mailbox.

NOTE

The first mailbox cannot be used for reception or transmission process.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Workaround possible but not implemented in the BSP, impacting functionality as described above.

ERR008650 GPMI: BCH ECC Randomizer will fail when reading an erased page

Description:

The BCH ECC has a randomizer module that is interfaced through the GPMI APBHDMA chain. The randomizer can generate random data based on BCH ECC encoded/decoded data. It can be employed to reduce the disturbances caused by a neighboring cell in the NAND chip, thus reducing bit errors. The randomizer should always be enabled when reading from MLC NAND flash memory under 20 nm and all TLC NAND flash memories.

Reading an erased page from NAND flash will always fail if the randomizer is enabled. The data in an erased page is all 0xFF.

If the page to be read is a programmed page, the data should be randomized before it is programmed into the flash. When the programmed page is read with the randomizer enabled, the BCH will do a de-randomize operation and produce program code with the correct data.

If the page to be read is an erased page, it should not be randomized and all data will 0xFF. If this page is read with the randomizer enabled, the BCH will also perform the de-randomize operation and produce incorrect raw data.

Projected Impact:

The randomizer when enabled will produce incorrect data only when reading an erased page of NAND flash memory. The issue does not occur when reading a programmed page. Devices are shipped with the Randomizer Enable fuse cleared (0) which will cause the randomizer to be disabled by default.

Workarounds:

Read a minimum subpage to determine if the page is programmed or erased before reading an entire page.

If the subpage is determined to be a programmed page, read the entire page with the randomizer enabled.

If the subpage is determined to be an erased page, read the entire page with the randomizer disabled.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Software workaround to be included in a future BSP release.

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 400kHz.

Projected Impact:

No failures have been observed when operating at 400 kHz. This errata 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.MX6 products:

I2C parent clock PERCLK_ROOT = 24M OSC

perclk_podf = 1

PERCLK_ROOT = 24M OSC/perclk_podf = 24MHz

I2C_IFDR = 0x2A

I2C clock frequency = 24MHz/64 = 375KHz

Proposed Solution:

No fix scheduled.

Linux BSP Status:

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

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 Freescale Linux BSP releases leave the ARCR_GUARD bits at the default value of 4'b0000.

ERR009636 MMDC: Random data corruption during reads from DDR memory

Description:

Random data corruption is observed on certain ICs when the MMDC DDR controller reads data from external DRAM (LPDDR2 or DDR3 memories). Due to a timing issue between the MMDC controller and the MMDC PHY, incorrect read command signals are sent to the external DDR memory. This can result in the mismatch of pointers in the DDR controller read FIFO, causing data corruption.

Projected Impact:

Random data corruption observed on reads from external DRAM (LPDDR2 or DDR3 memories) leading to various CPU exceptions.

Workarounds:

There are no software workarounds to resolve the issue. Increasing the VDD_SOC_CAP voltage can reduce the impact of this issue on affected ICs.

Proposed Solution:

Fixed in silicon revision 1.3.

Linux BSP Status:

No software workaround available.

ERR003747 PCIe: Reading the Segmented Buffer Depth Port Logic registers returns all zeros

Description:

When disabling the Dynamic Q Depth Adjustment, DBI reads to the Segmented Buffer Depth Port Logic registers return all zeros versus returning the hardwired default value. Internally, the DBI read access clears these registers, overwriting the default value with all zeros. Clearing these registers results in all zeros being returned for subsequent PCIe Cfg reads.

Following is an example scenario for this erratum:

1. Issue a PCIe Cfg read to any Port Logic Segmented Buffer Depth register.
   The read data value returned to the requester is the hardwired default value.
2. Issue a DBI read to same Port Logic Segmented Buffer Depth register.
   The read data value returned to the requester is all zeros.
3. Issue a PCIe Cfg read to same Port Logic Segmented Buffer Depth register.
   The read data value returned to the requester is all zeros.

Projected Impact:

The default Segmented Buffer Depth register values cannot be read when a DBI read access is performed with CX_DYNAMIC_SEG_SIZE = 0.

Workarounds:

PCIe Cfg, instead of DBI, should be used for reading the Segmented Buffer Depth Port Logic registers.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not applicable to the BSP

ERR005184 PCIe: Clock pointers can lose sync during clock rate changes

Description:

The digital to analog clock domain transfer of the frequency updates is susceptible to meta-stability errors as the transfer is done through a FIFO. During initial power-up, the FIFO ensures proper alignment by delaying the read pointer until the write clock has started. For correct operation of pointers, it is required that there are five edges of the write clock in two edges of the read clock.

When a rate change occurs from Gen1 to Gen2 or vice versa, the clock frequencies switch. During this switching, it is possible (depending upon internal clock tree delay in the PHY digital logic) that there are six write clock edges between two clock edges of the read clock. This causes the pointers to move out of sync and for some process/voltage/temperature corners can result in continuous corrupted reads of frequency update inputs to CDR phase mixer.

Once the pointers are misaligned, the condition will persist until the clocks are disabled or another phase shift occurs in clock phases as a result of rate change. The end result is the CDR loses lock in L0 state.

Projected Impact:

Low.

Workarounds:

From Cold start (LTSSM starts in Detect state):

- Disable MAC/LTSSM.
- Disable MAC/Gen2 support.
- Release MAC/LTSSM.
- Wait for MAC to enter L0.
- From L0, initiate MAC entry to Gen2 if EP/RC supports Gen2.
- Wait 2 ms (LTSSM timeout is 24 ms, PHY lock is ~5 µs in Gen2).
- If (MAC/LTSSM.state == Recovery.RcvrLock) && (PHY/rx_valid == 0), then pulse PHY/rx_reset. Transition to Gen2 is stuck.

Enter L2 from L0:

- Driver receives/requests entry to L2.
- Wait 2 ms (LTSSM timeout is 24 ms, PHY lock is ~10 µs in Gen1).
- If (MAC/LTSSM.state == Recovery.RcvrLock) && (PHY/rx_valid == 0), then pulse PHY/rx_reset. Transition to Gen1 is stuck.

Exit from L2 to L0:

- Driver receives/requests exit from L2.
- Disable MAC Gen2 support.
- Release LTSSM to wake-up from L2.
- Wait for entry to L0 and then repeat process of entering Gen2 from cold start case (going through Detect).

PHY reset process:

1. Disable RX in the PHY by CREG write: Address=16’h1005, Data=16’h0028
2. Enable RX in the PHY by CREG write: Address=16’h1005, Data=16’h0000

PHY rx_valid read:

- Sample rx_valid in the PHY by CREG read: Address=16’h100D, Data Mask=16’h0001 (rx_valid is bit 0)

MAC software registers:

1. Disable/enable LTSSM: write app_ltssm_enable.
2. Disable Gen2 (read/write link capability/status register):
3. Link in L0 event (driver should know this when the data link layer starts):
4. Request change to Gen2: (Cfg Directed Speed Change enabled. Write Gen2 Control Register DEFAULT_GEN2_SPEED_CHANGE).
5. Read LTSSM state (xmlh_ltssm_state[4:0].
6. Negotiate L2 entry/exit (software controlled D3).

Proposed Solution:

No fixed scheduled

Linux BSP Status:

Planned to be fixed in the Alpha BSP

ERR007520 PCIe: L1/L2 entry negotiation not restarted after interruption

Description:

Section 5.2 of the PCI Express base specification revision 3.0 states the following in relation to L1 and L2/L3 Ready entry: "If the negotiation is interrupted, for example by a trip through Recovery, the state machine in both components is reset back to the idle state." The core does not do this and only restarts the low-power entry negotiation process after both sides of the link enter electrical idle (EI) or on a "link down" event.

Scenario Setup:

This describes the scenario for L1 entry in an upstream port (USP); similar scenarios apply for L2/L3 Ready and in downstream ports (DSP):

- Enable L1 ASPM.
- Wait for L1 ASPM timer to expire.
- Wait until upstream port sends Active State L1 Enter DLLPs.
- Respond with positive Acknowledge.
- Inject bit errors on the link to trigger a transition to recovery.

Projected Impact:

The L1 entry process does not restart.

The DSP sends PM_Request_Ack DLLPs after exit from recovery until the USP restarts and successfully completes the L1 entry sequence for entering electrical idle (EI). TLP transmission will be blocked until the entry process completes.

The USP goes to EI immediately after exit from recovery instead of restarting the negotiation to enter the low-power state. The link will not resume normal operation until the DSP goes to EI.

Workarounds:

When supported, ensure that L0s is enabled. This improves the possibility of the core entering EI on either side of the link. Entering EI causes the power management state machines on both sides of the link to resynchronize.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR007554 PCIe: MSI Mask Register Reserved Bits not read-only

Description:

Per the PCI Specification, unimplemented mask register bits in the MSI capability should be reserved. All mask bits in the core are implemented with read-write attribute, regardless of the number of the number of MSI vectors requested. The PCI-SIG compliance test CFG 4.0.1 expects the reserved bits to be read-only and consequently fails if less than the maximum number of MSI vectors are requested by the core.

Projected Impact:

Low.

Workarounds:

Request maximum number of MSI vectors by writing the Multiple Message Capable field of the MSI Control Register with a value of 0x5 via the DBI register. Apply a mask on the writable mask bits in accordance with the value of cfg_multi_msi_cap.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround will be provided in a future BSP release.

ERR007555 PCIe: iATU—Optional programmable CFG Shift feature for ECAM is not correctly updating address (9000642041)

Description:

iATU Enabled (CX_INTERNAL_ATU_ENABLE =1) bit 28 (CFG_SHIFT_MODE) is enabled in the IATU_REGION_CTRL_2_OFF_OUTBOUND_0 register.

The implementation of the Enhanced Configuration Address Mapping (ECAM) feature violates the PCIe specification requirement that all reserved fields should always be "0". The basic idea is that the Bus/Device/Function (BDF) address is shifted 4 bits down so that the entire Cfg space can be mapped into a 256 MB region, rather than requiring multiple address translation tables, or a 4GB translation space. The BDF is then supposed to be shifted back up from bits 27:12 to 31:16 in the outgoing TLP (actually Bytes 8 and 9 in the Cfg TLP). The core does not do this translation correctly when the CFG Shift bit is set in an iATU entry.

Projected Impact:

The reserved bits 15:0 are not all "0" as required by the PCIe specification.

Workarounds:

None.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround can be implemented to mask or workaround this erratum. This erratum will result in impacted or reduced functionality as described above.

ERR007556 PCIe: Core Delays Transition From L0 To Recovery After Receiving Two TS OS And Erroneous Data (9000597455)

Description:

When the core is in L0 and receives two TS ordered sets followed by erroneous data, the core does not transition to Recovery immediately. The core will wait for the 128 us timeout and then move to Recovery if the core continuously receives erroneous data.

Scenario Setup:

Linkup to L0

Send two TS ordered sets to the core

Send some erroneous data to the core immediately

Continue sending erroneous data to the core for 128 us

Projected Impact:

Core delays transition to Recovery

Workarounds:

None

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround can be implemented to mask or workaround this erratum. This erratum will result in impacted or reduced functionality as described above.

ERR007557 PCIe: Extra FTS sent when Extended Synch bit is set (9000588281)

Description:

16-bit, 32-bit, or 64-bit PIPE I/F (CX_NB >= 2)

Gen1/Gen2 Mode (CX_GEN3_MODE = 2)

When running at Gen1 or Gen2 speed, if the valid core data width on a lane is 2s (two symbols) or higher and the Extended Synch bit is set in the Link Control Register, then the core sends one extra FTS (4097 instead of 4096) when exiting L0s.

Scenario Setup:

1. Set the Extended Synch bit in the Link Control Register.
2. Enable L0s ASPM by setting Link Control Register bit 0 to 1.
3. Bring the link to L0 at Gen1 or Gen2 speed and leave the link idle.
4. The controller goes to L0s after an L0s entry latency timeout.
5. Initiate a TLP transmission.
6. The controller wakes up and sends 4097 FTS.

Projected Impact:

There is no impact on a link in normal operating mode because the Extended Synch bit is used for external Link monitoring tools. It is not used in an operational PCIe Link.

When the core transmits FTS, the remote partner is in Rx_L0s.FTS. The next state for the remote partner is L0 if a SKP is received. If the Extended Synch bit is set, the core will transmit at least 12 separate SKP Ordered Sets during the 4096 FTSs transmitted. The PCIe Specification says that when the extended synch bit is set, the Receiver N_FTS timeout must be adjusted to no shorter than 40* [2048] * UI (2048 FTSs) and no longer than 40* [4096] * UI (4096 FTSs). The fact that the core sends 4097 FTSs instead of 4096 will not matter, because according to the requirement above, the remote partner will timeout before the extra FTS is sent.

Workarounds:

None

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround can be implemented to mask or workaround this erratum. This erratum will result in impacted or reduced functionality as described above.

ERR007559 PCIe: Core sends TS1 with non-PAD lane number too early in Configuration.Linkwidth.Accept State (9000574708)

Description:

When the downstream port (DSP) core enters Configuration.Linkwidth.Accept, it immediately starts sending TS1 with non-PAD lane number.

Projected Impact:

This might confuse the remote partner and lead to the formation of a narrower link.

Workarounds:

None

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround can be implemented to mask or workaround this erratum. This erratum will result in impacted or reduced functionality as described above.

ERR007573 PCIe: Link and lane number-match not checked in recovery (9000569433)

Description:

When the core's LTSSM is in Recovery.RcvLock or Recovery.RcvCfg, and it receives TS Ordered sets, it does not check whether the link and lane numbers of the received TS Ordered Sets match what is being transmitted on those same lanes.

Scenario Setup:

Core is in link state "Recovery.RcvLock" or "Recovery.RcvCfg" and receives TS Ordered Sets with link and lane number not matching what is being transmitted on those same Lanes.

The absence of link and lane number match checks in Recovery.RcvrLock and Recovery.RcvrCfg states only affects single lane configurations (CX_NL = 1). All configurations are not affect, as stated in the Impacted Configurations section above.

Projected Impact:

The core moves from Recovery.RcvLock to Recovery.RcvCfg or from Recovery.RcvCfg to Recovery.Idle and LTSSM misses the condition of link and lane number match and moves to the next state without robustness.

Workarounds:

None

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.

ERR007574 PCIe: Loopback—Core does not clear available credits when link goes down

Description:

In the loopback master, the core sees its own credits and not those of the link partner. If the link is directed into loopback master mode after it completes flow control initialization, and if it had advertised different credits to those of the remote link partner, then it will have an incorrect value for the remote link partner's credits. Normally a reset occurs after link down except in situations when the link is directed to loopback master mode.

Scenario Setup:

Ensure the link partner advertises different credit values to that of the core itself.

Enable the link to train to L0 and complete flow control initialization.

Direct the core to enter loopback as master.

Enable flow control initialization to complete in loopback. The credits available on the xadm_*_cdts outputs will not match those advertised by the receive path of the core, that is, the corresponding RADM_*Q_*CRD_VC0 parameter.

Projected Impact:

The core might have an incorrect credit count. This can cause hanging because of insufficient credits or it might cause the receive queue to overflow because of overstated credits.

Workarounds:

None

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround available.

ERR007575 PCIe: LTSSM delay when moving from L0 to recovery upon receipt of insufficient TS1 Ordered Sets (9000514662)

Description:

When the remote link partner enters Recovery.rcvrlock from the L0 state and transmits only two TS1 Ordered Sets (OS), the core can sometimes miss the second TS1 OS and therefore delay its entry into Recovery.rcvrlock.

Scenario Setup:

The remote link partner enters Recovery.rcvrlock from the L0 state and transmits only two TS1 OS's

The remote link partner then unusually moves to ElecIdle and de-asserts the PIPE signal rxvalid in Recovery.RcvrLock

The expected response from the core is that it will transition to Recovery.rcvrlock on receipt of the two TS1 OS's

The core receives a SKP OS or EIEOS that was inserted between the two TS1 Ordered Sets.

Note: This is an unusual verification setup, and in a real system the remote partner must keep sending TS1s in Recovery.RcvrLock and then core will move to Recovery after receiving 2 TS1s.

Projected Impact:

The core might miss the second TS1 OS because the remote partner only sent two TS1 OS's, the core will not receive a second TS1 OS and therefore stays in L0.

The PHY detects a decode error and passes it to the core.

The core sends the error message to the remote link partner.

The core does not get a response from the remote partner and replays the message three times.

The replay timer rolls over (caused by unacknowledged ERR_CORR messages) and a link retrain is requested.

The core moves to recovery.

Workarounds:

None

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.

ERR007577 PCIe: DLLP/TLP can be missed on RX path when immediately followed by EIOS (9000487440)

Description:

DLLP ACK frame is missed on the RX path. After the ACK, two EIOS are seen on the pipe interface. In this scenario, the ACK is missed by the RX logic, causing to the corresponding TLP to be re-transmitted from the TX replay buffer. Eventually, the link recovers from this event, as the receiver on the other side drops the re-transmitted TLP as a duplicate TLP. If the missed frame is a TLP, no ACK will be sent to the link partner, resulting in re-transmission of the TLP from the link partner.

Projected Impact:

Unnecessary re-transmission of a TLP.

Workarounds:

None

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.

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.

ERR007785 QuadSPI: Incorrect data returned in multiple flash use case when the read crosses flash boundary

Description:

In a multiple flash use case, QuadSPI returns incorrect data when a single read transaction crosses the boundary between the flashes.

In serial mode, a flash boundary can occur between flash A1 and A2, A2 and B1 and B1 and B2.

In parallel mode is will occur if a single read begins in A1-B1 pair and continues into the A2-B2 pair.

Projected Impact:

The QuadSPI can read incorrect data.

Workarounds:

Software should ensure that no read transaction crosses a flash boundary. This can be managed via the following methods:

- Reduce the data fetch amount of the AHB buffers to 64 bit so that no prefetch occurs. This will prevent any legal access from crossing the boundary.
- If prefetch is enabled then reserve a memory region, the size of the prefetch, prior to the boundary. So long as no accesses are made to this region of memory, no access will cross the boundary.
- Ensure that the core's cache is not prefetching across the boundary

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

ERR008611 QuadSPI: When an abort occurs to a suspended low-priority flash transaction during a new AHB request from the same low-priority master of a different address, then the new flash transaction starts from a wrong address.

Description:

When the HP_EN bit of QuadSPI_BUF0CR register is set high, then suspend-resume/ suspend-abort mechanisms occur to improve the read performance for the master associated with Buffer 0.

In one scenario, an AHB request from a high-priority master suspends an ongoing flash transaction (associated to a low-priority) and a flash transaction (associated to high-priority master) begins.

If the low-priority master (of which the transaction was suspended) positions another read AHB request exactly one bus clock cycle before the high-priority transaction completes, then the suspended transaction must abort, and the transaction for the new incoming address must begin. However, after the suspended transaction aborts, the buffer of that particular master does not flush, therefore; the buffer is hit for first few incoming addresses and begins a flash transaction at a latter address. This is not the correct behavior and can lead to the wrong data being read by the low-priority master.

Projected Impact:

The low-priority master can read incorrect data.

Workarounds:

Do not enable high-priority for Buffer 0 (leave QuadSPIx_BUF0CR[HP_EN] = 0).

Proposed Solution:

No fix scheduled.

Linux BSP Status:

No software workaround available.

ERR007637 RDC: Clearing violating address bit in MRVS (memory region violation status) register can only be from domain 0

Description:

For the QSPI resource domain protection using the resource domain controller (RDC), if domain zero is not on the access list for a QSPI memory region then the status register (MRVS) showing the first denied access attempt cannot be cleared. If domain zero is on the access list for a region then the status can be cleared by any bus master that can write to that MRVS register location.

Projected Impact:

The resource domain protection will work properly, but the MRVS status register cannot be cleared under the conditions listed above.

Workarounds:

If it is okay that all active resource domains are allowed to clear the MVRS register for a QSPI memory region then domain zero should be included on the allowed domains, regardless if domain zero is an active domain or not.

On the other hand, if it is preferred that the MRVS cannot be cleared by a bus master of one of the (perhaps offending) domains then the domains which need access to the memory region should not be assigned to domain zero and domain zero should not be included in the allowed access list for that memory region. The drawback is that no bus master of a domain can clear the register so only the first offending access will be captured. The offending address and offending domain of the first denied access will remain present in the MRVS until the next power cycle of the peripheral power domain.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround will be provided in a future BSP release.

ERR009572 RDC: Access to RDC registers will cause the CPU to hang if the PCIE_DISABLE fuse has disabled PCIe

Description:

On certain i.MX6 SoloX part numbers that do not support the PCIe module, Freescale programs the PCIe_DISABLE fuse to disable this module's functionality. When this PCIe_DISABLE fuse is programmed, it has the inadvertent effect of disabling a clock source required by the Resource Domain Controller (RDC) module on i.MX6 SoloX. The absence of this clock to the RDC causes the master (ARM Cortex A9 or M4) that initiates an RDC register access (read or write) to hang.

This issue is only observed on specific part numbers that have the PCIe_DISABLE fuse programmed to disable the PCIe module. The part numbers that support PCIe do not have this issue.

Projected Impact:

The ARM Cortext-A9 or ARM Cortext-M4 can hang.

Workarounds:

No software workarounds available that would allow RDC registers accesses on devices that have the PCIe_DISABLE fuse programmed to disable the PCIe module.

The issue only occurs when a master tries to access the RDC and has no impact if the application software does not access any of the RDC registers.

Proposed Solution:

For existing Rev 1.2 silicon devices that do not support the PCIe module, the PCIE_DISABLE fuse will not be programmed during the manufacturing process. These Rev 1.2 un-fused silicon devices have date code 1524 or later.

The date code can be read from the package markings. Directly below the part number, there is a code of the form xxxYYWW. YY is the year (15 for 2015) and WW is the work week number (24 for the 24th calendar week of that year).

Fixed in silicon revision 1.3

Linux BSP Status:

No software workaround available.

ERR008506 ROM: Incorrect NAND BAD Block Management

Description:

This issue occurs only when the first block in the firmware area (not FCB) is bad.

If the first block of firmware area is bad, then ROM will skip to the next block to get the first 4KB data. After reading the next data block, ROM returns to the first block (which was the bad block) and ECC checking fails. Afterward, ROM will go to secondary boot because it is a NAND boot device which supports secondary boot. So firmware2 will work in this case if a secondary boot image has been burned into NAND.

When the first block of the firmware area is bad, and the NAND page size is 4K or lower, this condition will occur. A bad block which is not the first one in firmware area will not cause this condition.

Projected Impact:

If the first block of the firmware area is bad, ECC checking of the NAND image may fail.

Workarounds:

1. Burn the correct firmware address in FCB to ensure the first block of firmware is not bad.
2. Burn firmware2 into NAND. The possibility of both the first block of firmware1 and the first block of firmware2 being bad is highly unlikely.

Proposed Solution:

No fix scheduled.

Linux BSP Status:

Workaround possible but not implemented in the BSP, impacting functionality as described above.

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 imx_3.10.53_1.1.0_ga.

ERR009059 USB: USB_OTG1_PWR (GPIO1_IO09) will output high during reset

Description:

USB_OTG1_PWR (GPIO1_IO09) will output high during the system reset period; it should be always low until USB drive enables this IO. During reset, GPIO1_IO09 configures to ALT6 and outputs the INT_BOOT value from the SRC.

Projected Impact:

This issue does not affect USB operation, but may impact USB certification testing. When system power is applied, the GPIO1_09/USB_OTG1_PWR signal will go high due and this causes leakage inside the i.MX 6SoloX. The result is the TA_VBUS_RISE timing can be much longer than 100 ms maximum the specification requires, causing a certification test to fail.

Workarounds:

Do not use GPIO1_09 as USB_OTG1_PWR for certification testing. USB_OTG1_PWR is available in two other alternate IOMUX configurations: ENET1_MDC [Alt 6] or QSPI1A_DATA2 [Alt 1].

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale 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.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 Freescale 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. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Document Number: IMX6SXCE
Rev. 1
04/2016