Document Number: IMX6SLLCE
Rev. 0.1, 09/2017

Chip Errata for the i.MX 6SLL

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

Table 1 provides a revision history for this document.

Table 1. Document Revision History

Rev. Number	Date	Substantive Changes
Rev. 0.1	09/2017	• Added a new erratum: – ERR011075
Rev. 0	04/2017	• Initial release

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

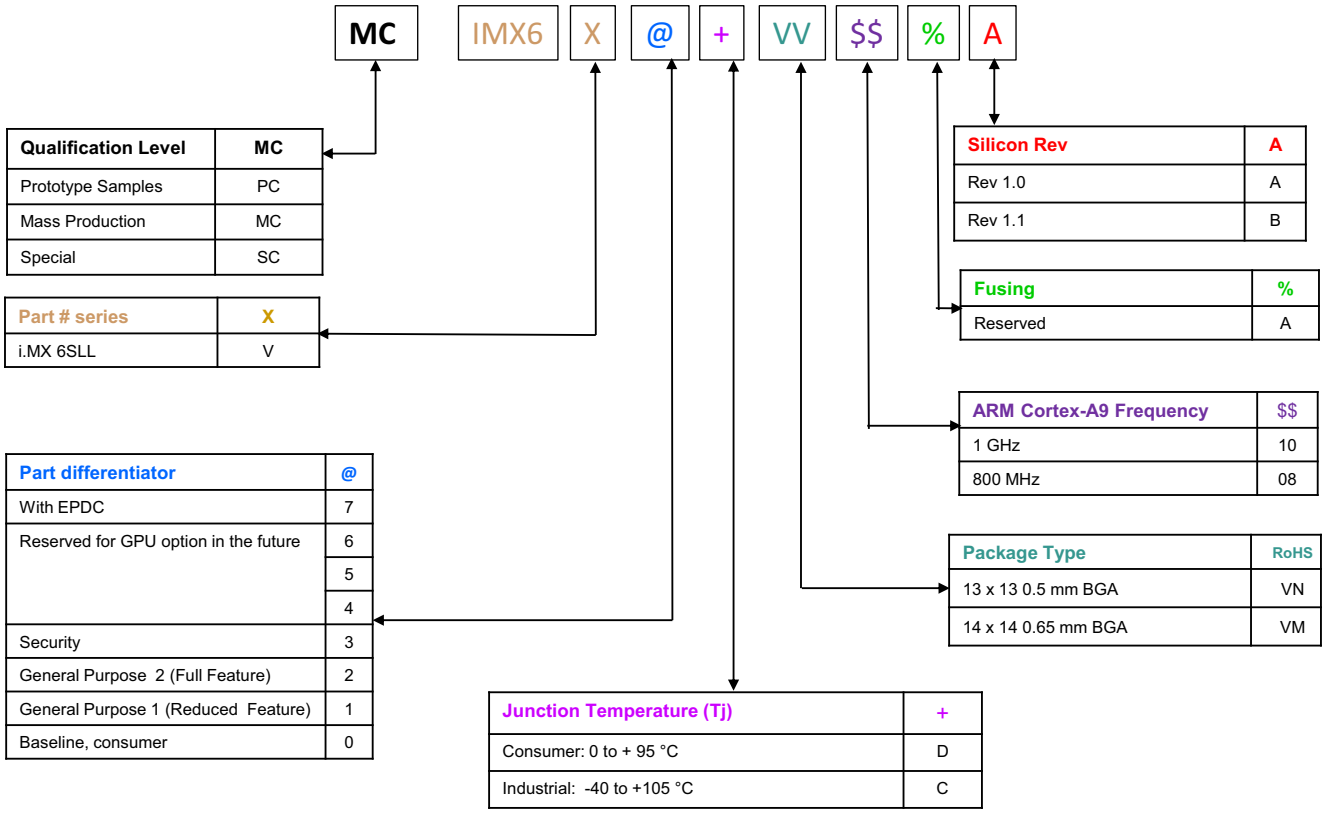

Figure 1. Revision Level to Part Marking Cross-Reference

For details on the Arm® configuration used on this chip (including Arm module revisions), please see the "Platform configuration" section of the "Arm Cortex®-A9 MPCore Platform" chapter of the *i.MX 6SLL Applications Processor Reference Manual* (IMX6SLLRM).

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

Table 2. Summary of Silicon Errata

Errata	Name	Solution	Page
	Arm®		
ERR003717	Arm: 740657—Global Timer can send two interrupts for the same event	No fix scheduled	5
ERR003723	Arm: 751476—May miss a watchpoint on the second part of an unaligned access that crosses a page boundary	No fix scheduled	7
ERR003724	Arm: 754322—Possible faulty MMU translations following an ASID switch	No fix scheduled	8
ERR003725	Arm: 725631—ISB is counted in Performance Monitor events 0x0C and 0x0D	No fix scheduled	10
ERR003726	Arm: 729817—MainID register alias address are not mapped on Debug APB interface	No fix scheduled	11
ERR003727	Arm: 729818—In debug state, next instruction is stalled when sdabort flag is set, instead of being discarded	No fix scheduled	12
ERR003732	Arm: 751471—DBGPCSR format is incorrect	No fix scheduled	13
ERR003734	Arm: 752519—An imprecise abort may be reported twice on non-cacheable reads	No fix scheduled	15
ERR003735	Arm: 754323—Repeated Store in the same cache line may delay the visibility of the store	No fix scheduled	16
ERR003736	Arm: 756421—Sticky Pipeline Advance bit cannot be cleared from debug APB accesses	No fix scheduled	18
ERR003737	Arm: 757119—Some “Unallocated memory hint” instructions generate an UNDEFINED exception instead of being treated as NOP	No fix scheduled	19
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	20
ERR003743	Arm/PL310: 754670—A continuous write flow can stall a read targeting the same memory area	No fix scheduled	21
ERR004326	Arm/MP: 761321—MRC and MCR are not counted in event 0x68	No fix scheduled	22
ERR004327	Arm/MP: 764319—Read accesses to DBGPRSR and DBGOSLSR may generate an unexpected UNDEF	No fix scheduled	23
ERR005183	Arm/MP: 771224—Visibility of Debug Enable access rights to enable/disable tracing is not ensured by an ISB	No fix scheduled	24
ERR005187	Arm/MP: 771223—Parity errors on BTAC and GHB are reported on PARTITYFAIL[7:6], regardless of the Parity Enable bit value	No fix scheduled	25
ERR005198	Arm/PL310: 780370—DATAERR, TAGERR, and Tag parity errors are incorrectly sampled by the eviction buffer, leading to data corruption	No fix scheduled	26
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	29
ERR007007	Arm/MP: 794073—Speculative instruction fetches with MMU disabled might not comply with architectural requirements	No fix scheduled	30

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	31
ERR009742	Arm: 795769—“Write Context ID” event is updated on read access	No fix scheduled	33
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	34
CCM			
ERR006223	CCM: Failure to resume from Wait/Stop mode with power gating	No fix scheduled	35
ERR007265	CCM: When improper low-power sequence is used, the SoC enters low power mode before the Arm core executes WFI	No fix scheduled	36
eCSPI			
ERR009535	eCSPI: Burst completion by SS signal in slave mode is not functional	No fix scheduled	37
MMDC			
ERR005778	MMDC: DDR Controller’s measure unit may return an incorrect value when operating below 100 MHz	No fix scheduled	38
SSI			
ERR003778	SSI: In AC97, 16-bit mode, received data is shifted by 4-bit locations	No fix scheduled	39
System Boot			
ERR011075	System Boot: Boot failures on certain devices when the boot image targets OCRAM	No fix scheduled	40
USB			
ERR004535	USB: USB suspend and resume flow clarifications	No fix scheduled	41
ERR006281	USB: Incorrect DP/DN state when only VBUS is applied	No fix scheduled	42
ERR010661	USB: VBUS leakage occurs if USBOTG1 VBUS is on and USBOTG2 VBUS transitions from on to off	No fix scheduled	43

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.

Projecting 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 following:

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.

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

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

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

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

Projected Impact:

The errata might cause MMU translation corruptions.

Workarounds:

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

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

The sequence:

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

becomes

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

the sequence:

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

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

becomes

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

and the sequence:

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

becomes

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround 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 NXP Linux BSP does not support this optional profiling feature. Users may add support for this profiling features as required.

ERR003726 Arm: 729817—MainID register alias address 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.

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.

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.

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

Description:

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

Conditions:

The erratum occurs when the following conditions are met:

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

Projected Impact:

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

Workarounds:

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

This is the behavior by default.

Proposed Solution:

No fix scheduled

Linux BSP Status:

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

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

Description:

In the Arm L2 cache controller, PL310, hazard checking is done on bits [31:5] of the address. When a read with Normal Memory (cacheable or not) attributes is received by PL310, hazard checking is performed with the active writes of the store buffer. If an address matching is detected, the read is stalled till the write completes.

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

Conditions:

The erratum occurs when the following conditions are met:

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

Projected Impact:

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

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

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

Workarounds:

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

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

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

Description:

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

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

Projected Impact:

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

Workarounds:

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The NXP Linux BSP does not support this optional profiling feature. Users may add support for this profiling feature as required.

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.

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 encountered in normal device operation. Users should use Arm recommended workaround if using this debug trace feature.

ERR005187 Arm/MP: 771223—Parity errors on BTAC and GHB are reported on PARTITYFAIL[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 mispredicted, 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 not that the i.MX 6 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 NXP Linux implementation, the parity error detection is disabled and GIC parity interrupt 125, is masked in the BSP. The parity feature is disabled by default and should not be enabled.

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

Description:

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

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

Projected Impact:

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

Workarounds:

There is no workaround to this erratum.

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround not implemented in Linux BSP. Functionality or mode of operation in which the erratum may manifest itself is not used. The NXP Linux BSP does not support this optional profiling feature. Users may add support for this profiling feature as required.

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 enable when it performs BTAC flush in LOPM 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 NXP 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 NXP 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.

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

Description:

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

Projected Impact:

Device might fail to resume from low-power state.

Workarounds:

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

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

RBC_COUNT × (1/32K RTC Frequency) ≥ (25 + PDNSCR_SW2ISO) × (1/IPG_CLK Frequency)
PDNSCR_ISO2SW = PDNSCR_ISO = 1 (counts in IPG_CLK clock domain)

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in Linux BSP codebase starting in release GA L3.0.35_1.1.0.

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 wakes up from an interrupt event, which is masked by GPC or not visible to GPC, such as an interrupt from a local timer.
4. Set CCM_CLPCR[1:0] to 2'b01 or 2'b10.
5. Arm core executes WFI.

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

Projected Impact:

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

Workarounds:

Software workaround:

1. Software should trigger IRQ #32 (IOMUX) to be always pending by setting IOMUX_GPR1_GINT.
2. Software should then unmask IRQ #32 in GPC before setting CCM Low-Power mode.
3. Software should mask IRQ #32 right after CCM Low-Power mode is set (set bits 0-1 of CCM_CLPCR).

Proposed Solution:

No fix scheduled

Linux BSP Status:

Software workaround implemented in Linux BSP codebase. A patch is included in both BSP kernels v3.10.9 and v3.0.35.

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.

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

Description:

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

Projected Impact:

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

Workarounds:

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

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

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

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

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

Description:

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

Projected Impact:

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

Workarounds:

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

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

ERR011075 System Boot: Boot failures on certain devices when the boot image targets OCRAM

Description:

A boot image corruption can result in a boot failure for certain boot devices under specific conditions as described below. The boot failure only occurs if all conditions below are satisfied.

Conditions:

There are four specific conditions resulting in this boot failure.

- i.MX 6SLL silicon revision 1.1 device shipped prior to date code 1732
- The boot device is SD/eMMC/SPINOR
- Boot image runs from internal memory—On Chip RAM (OCRAM) space
- Start address of the boot image in OCRAM is between 0x907000 and 0x908000

Projected Impact:

Boot image corruption leading to boot failure and possible entry into Serial Downloader mode.

Workarounds:

User should set the boot image start address greater or equal to 0x908000 (if the boot image running in OCRAM) to prevent the boot failure when using the SD/eMMC/SPINOR boot devices. The impact of this workaround is a reduction in the usable memory region by 4 KB. Alternatively, users can use a boot image load address in the external DDR memory instead of the internal OCRAM.

Proposed Solution:

Fixed in i.MX 6SLL silicon revision 1.1 devices shipped after date code 1732.

Linux BSP Status:

Workaround not implemented in BSP; No impact to NXP BSP users, since the default BSP boot image uses external DDR as the load address and not internal OCRAM.

ERR004535 USB: USB suspend and resume flow clarifications

Description:

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

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

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

Projected Impact:

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

Workarounds:

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

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

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

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

Description:

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

Projected Impact:

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

Workarounds:

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

Proposed Solution:

No fix scheduled

Linux BSP Status:

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

ERR010661 USB: VBUS leakage occurs if USBOTG1 VBUS is on and USBOTG2 VBUS transitions from on to off

Description:

When two USB ports work as OTG or device simultaneously. One VBUS (selected by PMU_REG_3P0.vbus_sel bit) voltage will not drop after cable unplug, causing the port to fail to detect the cable detach. If these two ports do not need to support detach detection, simultaneously using two OTGs or devices can be supported.

Conditions:

When two USB ports work as OTGs or devices simultaneously.

Projected Impact:

Do not use two OTGs or devices simultaneously. Only four scenarios are supported:

- One for OTG/Device, another for Host.
- One for OTG/Device, another is un-used.
- One for Host, another for Host.
- One for Host, another is un-used.

Workarounds:

Only one port can be used as OTG or device. The other port must be used as host. Set the PMU_REG_3P0.vbus_sel bit to select the host port.

Proposed Solution:

No fix scheduled

Linux BSP Status:

No software workaround available

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

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

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

Document Number: IMX6SLLCE
Rev. 0.1
09/2017