MPC8536E and MPC8535E Chip Errata

This document details all known silicon errata for the MPC8536 and MPC8535 devices. The following table provides a revision history for this document.

Table 1. Revision History

Revision	Date	Significant Changes
4	09/2011	• Added USB-A005, USB-A007 • Modified CPU 2, SEC-A001, USB 9, and USB-A001
3	12/2010	• Updated USB-A002, USB-A003 • Added SEC-A001, CPU-A005 • Modified CPU 1 impact
2	07/2010	• Added DMA-A001, eLBC-A001, eSPI-A001, eTSEC-A002, IEEE1588-A001, PCIe-A001, PCI 2, SATA-A002, and USB-A003 • Modified ESDHC-A001 and DDR 4 workaround • Modified SATA 1 and ESDHC 17 • In eTSEC 19, added: "When using packets larger than 2700 bytes, it is recommended to turn TOE off. " to workaround section. • In SATA 2, headline revised to "SATA controller hangs when handling data integrity errors." It had been "Handling Data Integrity Errors" • In PCIe 2, changed Fix plan to "Fixed in Rev 1.2" • Added Silicon Revision 1.2 to Table 2 and Table 3.
1	10/2009	• Added CPU-A001. • Changed DDR7 Description 4th sentence to read: "However, some DIMMs will be routed such that these data bits will not be connected to DQ[0] of each discrete device." • Changed DDR3 Headline to read: "MCKE signal may not function correctly at assertion of $\overline{\text{HRESET}}$." • Changed DDR3 Description 1st sentence to read: "During the assertion of $\overline{\text{HRESET}}$ (excluding the initial power on reset) ..."
0	09/2009	• Initial release

Table 2 provides a cross-reference to match the revision code in the processor version register to the revision level marked on the device.

Table 2. Revision Level to Part Marking Cross-Reference

Part	Revision	e500 V2 Core Revision	Processor Version Register Value	System Version Register Value
MPC8536E	1.2	3.0	0x8021_0030	With Security: 0x803F_0092
MPC8536	1.2	3.0	0x8021_0030	Without Security: 0x8037_0092
MPC8535E	1.2	3.0	0x8021_0030	With Security: 0x803F_0192
MPC8535	1.2	3.0	0x8021_0030	WithoutSecurity: 0x8037_0192
MPC8536E	1.1	3.0	0x8021_0030	With Security: 0x803F_0091
MPC8536	1.1	3.0	0x8021_0030	Without Security: 0x8037_0091
MPC8535E	1.1	3.0	0x8021_0030	With Security: 0x803F_0191
MPC8535	1.1	3.0	0x8021_0030	WithoutSecurity: 0x8037_0191
MPC8536E	1.0	3.0	0x8021_0030	With Security: 0x803F_0090
MPC8536	1.0	3.0	0x8021_0030	Without Security: 0x8037_0090
MPC8535E	1.0	3.0	0x8021_0030	With Security: 0x803F_0190
MPC8535	1.0	3.0	0x8021_0030	Without Security: 0x8037_0190

Table 3 summarizes all known errata and lists the corresponding silicon revision level to which it applies. A ‘Yes’ entry indicates the erratum applies to a particular revision level, while a ‘No’ entry means it does not apply.

Table 3. Summary of Silicon Errata and Applicable Revision

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	1.2
	CPU				
CPU 1	"mbar MO = 1" instruction fails to order caching-inhibited guarded loads and stores	No plans to fix	Yes	Yes	Yes
CPU 2	Single-precision floating-point zero value may have the wrong sign	No plans to fix	Yes	Yes	Yes
CPU 3	Branch Fails to Decrement CTR in Presence of D-cache Parity Error	No plans to fix	Yes	Yes	Yes
CPU 4	Incorrect Value May be Loaded Into SRR0 on ITLB Miss When Executing **icbtls**	No plans to fix	Yes	Yes	Yes
CPU 5	A performance monitor time base transition event will not freeze the performance monitor counters and may not cause an exception	No plans to fix	Yes	Yes	Yes
CPU-A001	Flash-Invalidation of TLB1 with "mtspr MMUCSR0" may fail	No plans to fix	Yes	Yes	Yes
CPU-A005	Enabling IEEE 754 exceptions can cause errors	No plans to fix	Yes	Yes	Yes
	DDR				
DDR 1	A false address parity error may be detected when DDR_SDRAM_CFG[MEM_EN] is set	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	1.2
DDR 2	The memory controller does not use the correct default impedances for the IOs in DDR3 mode	No plans to fix	Yes	Yes	Yes
DDR 3	MCKE signal may not function correctly at assertion of $\overline{\text{HRESET}}$	No plans to fix	Yes	Yes	Yes
DDR 4	DDR controller may not work with dual-ranked DDR3 registered DIMM if write leveling is enabled	No plans to fix	Yes	Yes	Yes
DDR 5	DDR to CCB clock ratios of greater than 1.5:1 can result in DDR configuration register corruption	Fixed in Rev. 1.1	Yes	No	No
DDR 6	Asserting panic interrupt via $\overline{\text{IRQ\_OUT}}$ the DDR controller may violate the JEDEC specifications	No plans to fix	Yes	Yes	Yes
DDR 7	The DDR controller may fail the write leveling sequence for DDR3	No plans to fix	Yes	Yes	Yes
DDR 8	Data may become corrupted for DDR3 32-bit bus mode	No plans to fix	Yes	Yes	Yes
DMA					
DMA-A001	Uncorrectable data read errors for DMA transfers may also cause DMA lockup or additional detectable data corruption	No plans to fix	Yes	Yes	Yes
DUART					
DUART 1	BREAK detection triggered multiple times for a single break assertion	No plans to fix	Yes	Yes	Yes
eLBC					
eLBC1	Multi-bank DRAM and SDRAM do not work without any external logic	No plans to fix	Yes	Yes	Yes
eLBC 2	LTEATR and LTEAR may show incorrect values under certain scenarios	No plans to fix	Yes	Yes	Yes
eLBC-A001	Simultaneous FCM and GPCM or UPM operation may erroneously trigger bus monitor timeout	No plans to fix	Yes	Yes	Yes
eSPI					
eSPI-A001	Master out slave in (MOSI) master data output hold time tNIKHOX should be 3.3 ns for temperatures less than 0° C	No plans to fix	Yes	Yes	Yes
eTSEC					
eTSEC 1	Frame is dropped with collision and HALFDUP[Excess Defer] = 0	No plans to fix	Yes	Yes	Yes
eTSEC 2	Fetches with errors not flagged, may cause livelock or false halt	No plans to fix	Yes	Yes	Yes
eTSEC 3	Multiple BD frame may cause hang	No plans to fix	Yes	Yes	Yes
eTSEC 4	VLAN Insertion corrupts frame if user-defined Tx preamble enabled	No plans to fix	Yes	Yes	Yes
eTSEC 5	User-defined Tx preamble incompatible with Tx Checksum	No plans to fix	Yes	Yes	Yes
eTSEC 6	Transmit fails to utilize 100% of line bandwidth	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	1.2
eTSEC 7	Unexpected babbling receive error in FIFO modes	No plans to fix	Yes	Yes	Yes
eTSEC 8	ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software	No plans to fix	Yes	Yes	Yes
eTSEC 9	Controller stops transmitting pause control frames	Fixed in Rev. 1.1	Yes	No	No
eTSEC 10	Half-duplex collision on FCS of Short Frame may cause Tx lockup	No plans to fix	Yes	Yes	Yes
eTSEC 11	Magic Packet Sequence Embedded in Partial Sequence Not Recognized	No plans to fix	Yes	Yes	Yes
eTSEC 12	MAC: Malformed Magic Packet Triggers Magic Packet Exit	No plans to fix	Yes	Yes	Yes
eTSEC 13	No parser error for packets containing invalid IPv6 routing header packet	No plans to fix	Yes	Yes	Yes
eTSEC 14	Receive pause frame with PTV = 0 does not resume transmission	No plans to fix	Yes	Yes	Yes
eTSEC 15	TxBD polling loop latency is 1024 bit-times instead of 512	No plans to fix	Yes	Yes	Yes
eTSEC 16	MAC: Rx frames of length MAXFRM or MAXFRM-1 are marked as truncated	No plans to fix	Yes	Yes	Yes
eTSEC 17	Misfiled Packets Due to Incorrect Rx Filer Set Mask Rollback	No plans to fix	Yes	Yes	Yes
eTSEC 18	Incorrect frame data in L3+L4-only or L4-only parsing for FIFO mode	No plans to fix	Yes	Yes	Yes
eTSEC 19	Excess delays when transmitting TOE=1 large frames	No plans to fix	Yes	Yes	Yes
eTSEC 20	Controller may not be able to transmit pause frame during pause state	No plans to fix	Yes	Yes	Yes
eTSEC-A001	MAC: Pause time may be shorter than specified if transmit in progress	No plans to fix	Yes	Yes	Yes
eTSEC-A002	Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame	No plans to fix	Yes	Yes	Yes
IEEE1588					
IEEE1588_1	TxPAL timestamp uses TxBD snoop enable instead of Tx data	No plans to fix	Yes	Yes	Yes
IEEE1588_2	Odd prescale values not supported	No plans to fix	Yes	Yes	Yes
IEEE1588_3	Tx FIFO data parity error (DPE) may corrupt Tx timestamps if TMR_CTRL[TRTPE]=1	No plans to fix	Yes	Yes	Yes
IEEE1588_4	eTSEC 1588: Write to reserved 1588 register space causes system hang	No plans to fix	Yes	Yes	Yes
IEEE1588_5	1588 alarm fires when programmed to less than current time	No plans to fix	Yes	Yes	Yes
IEEE1588_6	IEEE 1588 accuracy can be adversely impacted in systems using multiple unsynchronized gigabit Ethernet ports	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	1.2
IEEE1588-A001	Incorrect received timestamp or dropped packet when 1588 time-stamping is enabled	No plans to fix	Yes	Yes	Yes
	GPIO				
GPIO 1	Invalid data read from GPDAT bits corresponding to pins configured as outputs	Fixed in Rev. 1.1	Yes	No	No
	I2C				
I2C 1	Enabling I^{2}C could cause I^{2}C bus freeze when other I^{2}C devices communicate	No plans to fix	Yes	Yes	Yes
	PEX				
PCIe 1	Reads to PCI Express CCSRs or local config space temporarily return all Fs	No plans to fix	Yes	Yes	Yes
PCIe 2	Excess correctable errors in receiving DLLPs and TLPs on x2 link	Fixed in Rev 1.2	Yes	Yes	No
PCIe-A001	PCI Express Hot Reset event may cause data corruption	No plans to fix	Yes	Yes	Yes
	PCI				
PCI 1	Master-Abort issued incorrectly for outbound DAC with subtractive decode	No plans to fix	Yes	Yes	Yes
PCI 2	Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang	No plans to fix	Yes	Yes	Yes
	SATA				
SATA 1	Reads of one sector from hard disk may return less data than requested	No plans to fix	Yes	Yes	Yes
SATA 2	SATA controller hangs when handling data integrity errors	No plans to fix	Yes	Yes	Yes
SATA 3	SATA BIST Activate FIS L bit is only valid in combination with the T bit	Fixed in Rev. 1.1	Yes	No	No
SATA 4	SATA Host does not acknowledge BIST Activate FIS and it immediately enters loopback mode.	No plans to fix	Yes	Yes	Yes
SATA 5	BIST-L mode is not enabled by BIST-L FIS	Fixed in Rev 1.1	Yes	No	No
SATA 6	SATA: DMAT handling in SATA not as expected	No plans to fix	Yes	Yes	Yes
SATA-A002	ATAPI commands may fail to complete	No plans to fix	Yes	Yes	Yes
	SEC				
SEC 1	Non-compliant implementation of deterministic pseudo-random number generator	No plans to fix	Yes	Yes	Yes
SEC 2	Kasumi hardware ICV checking does not work	No plans to fix	Yes	Yes	Yes
SEC-A001	Channel Hang with Zero Length Data	No plans to fix	Yes	Yes	Yes
	USB				
USB 1	Read of PERIODICLISTBASE after successive writes may return a wrong value in host mode	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	1.2
USB 2	SE0_NAK issue	No plans to fix	Yes	Yes	Yes
USB 3	NackCnt field is not decremented when received NYET during FS/LS Bulk/Interrupt mode	No plans to fix	Yes	Yes	Yes
USB 4	When an ACK or NAK is sent from the device in response to a PING, the CERR counter value is not being reset to the initial value	No plans to fix	Yes	Yes	Yes
USB 5	In device mode, when receiving a Token OUT, if a Rx flush command is issued at the same time, to the same endpoint, the packet will be lost	No plans to fix	Yes	Yes	Yes
USB 6	In host mode, when the software forces a port resume by writing into the FPR bit of the portsc register, the port change detect interrupt bit is falsely fired	No plans to fix	Yes	Yes	Yes
USB 7	Transmit data loss based on bus latency	No plans to fix	Yes	Yes	Yes
USB 8	Priming ISO over SOF will cause transmitting bad packet with correct CRC	No plans to fix	Yes	Yes	Yes
USB 9	Missing SOFs and false babble error due to Rx FIFO overflow	No plans to fix	Yes	Yes	Yes
USB 10	No error interrupt and no status will be generated due to ISO mult3 fulfillment error	No plans to fix	Yes	Yes	Yes
USB 11	Wrong value read in BURSTSIZE[TXPBURST] and BURSTSIZE[RXPBURST] registers	No plans to fix	Yes	Yes	Yes
USB 12	CRC not inverted when host under-runs on OUT transactions	No plans to fix	Yes	Yes	Yes
USB 13	Core device fails when it receives two OUT transactions in a short time	No plans to fix	Yes	Yes	Yes
USB 14	USB Controller locks after Test mode "Test_K" is completed	No plans to fix	Yes	Yes	Yes
USB 15	NAK counter decremented after receiving a NYET from device	No plans to fix	Yes	Yes	Yes
USB-A001	Last read of the current dTD done after USB interrupt	No plans to fix	Yes	Yes	Yes
USB-A002	Device does not respond to INs after receiving corrupted handshake from previous IN transaction	No plans to fix	Yes	Yes	Yes
USB-A003	Illegal NOPID TxCmd issued by USB Controller with ULPI Interface	No plans to fix	Yes	Yes	Yes
USB-A005	ULPI Viewport not Working for Read or Write Commands With Extended Address	No plans to fix	Yes	Yes	Yes
USB-A007	Host controller fails to enter the PING state on timeout during High Speed Bulk OUT/DATA transaction	No plans to fix	Yes	Yes	Yes
eSDHC					
ESDHC 1	Read/Write Multiple command when block count is 1	No plans to fix	Yes	Yes	Yes
ESDHC 2	Force Event Register	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	1.2
ESDHC 3	eSDHC reads more data than expected from the system	No plans to fix	Yes	Yes	Yes
ESDHC 4	eSDHC indicates AUTO CMD12 interrupt later than expected	No plans to fix	Yes	Yes	Yes
ESDHC 5	Data corruption during write with pause operation	No plans to fix	Yes	Yes	Yes
ESDHC 6	Cannot initiate non-data command while data transfer is in progress	No plans to fix	Yes	Yes	Yes
ESDHC 7	Incorrect data is written to a card after transfer paused	No plans to fix	Yes	Yes	Yes
ESDHC 8	CRC might be corrupted for write data transfer if it is preceded by read transfer	Fixed in Rev. 1.1	Yes	No	No
ESDHC 9	Data End Bit Error (DEBE, bit 22 in the interrupt status register) incorrectly set if Card Interrupt is driven in SDIO 4-bit mode	No plans to fix	Yes	Yes	Yes
ESDHC 10	During multi-write operation, unexpected Transfer Complete (IRQSTAT[TC] register) bit can be set	No plans to fix	Yes	Yes	Yes
ESDHC 11	eSDHC interrupt not negated in case of card insertion/ removal	No plans to fix	Yes	Yes	Yes
ESDHC 12	Unable to issue CMD12 if SD clock shuts off due to slow system access	No plans to fix	Yes	Yes	Yes
ESDHC 13	Manual Asynchronous CMD12 abort operation causes protocol violations	No plans to fix	Yes	Yes	Yes
ESDHC 14	PRSSTAT[CIDHB] is not reliable for commands with busy (R1b)	No plans to fix	Yes	Yes	Yes
ESDHC 15	The $\overline{\text{SDHC\_WP}}$ signal polarity is reversed	Fixed in Rev 1.1	Yes	No	No
ESDHC 16	eSDHC cannot finish write operation after continuing from Block Gap Stop	No plans to fix	Yes	Yes	Yes
ESDHC 17	Corrupted data read from eSDHC for certain values of WML[RD_WML] and BLKATTR[BLKSIZE]	No plans to fix	Yes	Yes	Yes
ESDHC 18	Invalid generation of Block Gap Event	No plans to fix	Yes	Yes	Yes
ESDHC 19	Invalid IRQSTAT[TC] is set for synchronous abort	No plans to fix	Yes	Yes	Yes
ESDHC 20	BLKATTR[BLKCNT] is not reliable for Block Gap Stop when BLKATTR[BLKZSE] ≤ 4	No plans to fix	Yes	Yes	Yes
ESDHC-A001	Data timeout counter (SYSCTL[DTOCV]) is not reliable for values of 0x4, 0x8, and 0xC	No plans to fix	Yes	Yes	Yes
PMC					
PMC 1	Cannot wake up from Jog mode	Fixed in Rev 1.1	Yes	No	No
PMC 2	PMRCCR[RCNT_PRE] register reset value is incorrect	Fixed in Rev 1.1	Yes	No	No
JTAG					
JTAG 1	The Debugger cannot read and write to the DDR SDRAM or local bus memory-mapped regions during the deep sleep mode	Fixed in Rev 1.1	Yes	No	No

Table continues on the next page...

Table 3. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	1.2
JTAG 2	JTAG debugger cannot access L2 cache as memory-mapped SRAM	Fixed in Rev 1.1	Yes	No	No

CPU 1: "mbar MO = 1" instruction fails to order caching-inhibited guarded loads and stores

Description: This errata describes a failure of the e500 **mbar** instruction when the MO field is one. In particular, the "**mbar** MO = 1" instruction fails to act as a barrier which cannot be bypassed by caching-inhibited loads.

Assume the following instruction sequence:

- **stw** caching-inhibited, guarded address A
- **mbar** MO = 1
- **lwz** caching-inhibited, guarded address B

The “**mbar** MO = 1” instruction is intended to be a barrier which prevents the **lwz** from executing before the **stw** has been performed. However, the e500 does not behave as intended, and allows the **lwz** to be executed before the **stw** has been performed.

Impact: This errata is most likely to affect device drivers that depend on “**mbar** MO = 1” to ensure that the effects of caching-inhibited stores are seen by a device before a subsequent caching-inhibited guarded load is executed.

The “**mbar** MO = 1” instruction is intended to order:

- Cacheable stores
- Cache-inhibited loads
- Cache-inhibited stores

The only case where the instruction may not behave correctly is where cache-inhibited loads may erroneously bypass cache-inhibited stores.

Workaround: Use "**mbar** MO = 0" to ensure that caching-inhibited guarded loads do not bypass the memory barrier.

Fix plan: No plans to fix

CPU 2: Single-precision floating-point zero value may have the wrong sign

Description:	When performing single-precision floating-point operations that produce a result of zero, the sign of the zero value may be incorrect.
Impact:	Single-precision floating-point operations that result in zero may not be compatible with IEEE Std 754™.
Workaround:	Use double-precision floating-point
Fix plan:	No plans to fix

CPU 3: Branch Fails to Decrement CTR in Presence of D-cache Parity Error

Description: There is a potential confluence of events that results in a failure to decrement the CTR when the branch instruction is executed. The necessary conditions are as follows:

- The branch is mispredicted, which can occur if branch prediction is either of the following:
  - Enabled
  - Disabled, and the branch is actually taken
- There is a load instruction in the branch's mispredicted path, and that load hits in the L1 D-Cache, and the cache line has a parity error.
- The parity error is detected within a narrow timing window of when the branch is deallocated.

Under these conditions, the branch redirects the instruction stream to the correct branch target, but the CTR is not decremented.

The speculative load that hits the parity error is flushed when the branch is resolved. D-Cache parity errors cause precise machine check interrupts. Therefore, because the load instruction is flushed, no machine check occurs.

After the mispredicted branch is resolved, the core begins executing from the corrected path. However, if an interrupt occurs during the execution of one of the first few instructions in the corrected path, there is a possibility that SRR0 points to an instruction in the mispredicted path.

Impact: This errata may cause undetected, erroneous program behavior in the presence of L1 D-Cache parity errors.

Workaround: None

Fix plan: No plans to fix

CPU 4: Incorrect Value May be Loaded Into SRR0 on ITLB Miss When Executing icbtls

Description: When executing the **icbtls**, if the instruction fetch unit fetches a line that results in an ITLB Miss interrupt, the value loaded into SRR0 will be incorrect.

Impact: This failure could cause the handler to return (**rfi**) to the wrong address.

This errata does not apply to **icbtls** instructions if CT ! = 0.

Workaround: This problem can be avoided by executing **isync** after **icbtls**. The program must ensure that no ITLB Miss can occur between the execution of the **icbtls** and the execution of the **isync**. This implies that the **isync** should reside on the same page as the **icbtls**.

A sequence of **icbtls** instructions can be followed by a single **isync** to avoid this problem. No ITLB Miss exceptions should be allowed between the execution of the first **icbtls** in this sequence and the **isync** at the end of the sequence.

There is no need to avoid asynchronous interrupts between the execution of **icbtls** and the following **isync**. The occurrence of the interrupt has the same effect as the **isync** and prevents the problem.

Fix plan: No plans to fix

CPU 5: A performance monitor time base transition event will not freeze the performance monitor counters and may not cause an exception

Description: The performance monitor counters will not freeze when a time base transition occurs even though PMGC0[TBEE] and PMGC0[FCECE] are enabled. Also, a performance monitor exception may not happen when a time base transition occurs even though PMGC0[TBEE] and PMGC0[PMIE] are enabled.

Impact: Performance monitor information about processor activity cannot be gathered during a precise interval based on the time base timer.

Workaround: The counters freeze when the msb = 1 in PMCx, PMLCax[CE] = 1, and PMGC0[FCECE] is enabled. Exceptions occur when the msb = 1 in PMCx, PMCLCax[CE] = 1, and PMGC0[PMIE] is enabled. Therefore, one of the performance monitor counters can be set to count a specific number of processor cycles that corresponds to the time interval desired.

After calculating the specific number of processor cycles required, program PMCx to 0x8000_0000 minus this number. This causes PMCx to overflow after the desired amount of cycles.

Fix plan: No plans to fix

CPU-A001: Flash-Invalidation of TLB1 with "mtspr MMUCSR0" may fail

Description: The e500v1 and v2 specification says that writing a 1 to MMUCSR0[TLB1_FI] will flash-invalidate all TLB1 entries that are not marked with "IPROT". However, this flash-invalidate mechanism may fail if the execution of the **mtspr** loosely coincides with the execution of a **tlbivax** that invalidates entries in the TLB1 (i.e., a **tlbivax** with bits 60–61 of the effective address set to binary 10). This **tlbivax** may be executed either on the same processor or on a different processor than the "**mtmsr MMUCSR0**" instruction.

Impact: Failure to invalidate all of the intended lines can result in operating system failure. In most operating systems, invalidation of TLB entries is restricted to a couple of places in the OS, and they are usually performed under controlled circumstances. Therefore, this bug may not impact a particular OS.

Workaround: Use one of the following options:

- Always invalidate the entire TLB1 with **tlbivax** with bits 60–61 = 11 instead of "**mtspr MMUCSR0**".
- Ensure that **tlbivax** and "**mtspr MMUCSR0**" cannot be executed simultaneously by using mutual exclusion locks around any code that invalidates TLB1.
- In a single-core environment, it is sufficient to ensure that **tlbivax** does not closely follow the **mtspr** that sets MMUCSR0[TLB1_FI] = 1 without an intervening **msync**.

Fix plan: No plans to fix

A-001428(CPU-A005): Enabling IEEE 754 exceptions can cause errors

Affects: CPU

Description: This issue can occur if a single-precision floating-point, double-precision floating-point, or vector floating-point instruction on a mispredicted branch path signals one of the floating-point data interrupts enabled by the SPEFSCR (FINVE, FDBZE, FUNFE or FOVFE bits). This interrupt must be recorded in a one-cycle window when the misprediction is resolved.

If this extremely rare event should occur, the result could be that the SPE Data Exception from the mispredicted path may be reported erroneously if a single-precision floating-point, double-precision floating-point, or vector floating-point instruction is the second instruction on the correct branch path.

It is only possible for this erratum to occur if any of the SPEFSCR exception enable bits (FINVE, FDBZE, FUNFE or FOVFE) are set to one.

Impact: A correctly executing floating point instruction that is the second instruction on the correct path may take an unexpected data exception. This is caused by an un-related floating point instruction that has been cancelled on the mis-predicted path.

Workaround: Use one of the following options:

- Ensure that the floating-point data exceptions are disabled by clearing the SPEFSCR exception enable bits (FINVE, FDBZE, FUNFE or FOVFE).
- Have the exception handler make the hardware re-execute the instruction, if a floating point instruction causes an unexpected data exception. If the exception was a result of this erratum, there will be no exception on re-execution. Freescale will make this modification to the exception handler we provide to the open source community.

Fix plan: No plans to fix

DDR 1: A false address parity error may be detected when DDR_SDRAM_CFG[MEM_EN] is set

Description: Address parity is enabled by setting DDR_SDRAM_CFG_2[AP_EN] before the memory controller has been enabled. When DDR_SDRAM_CFG[MEM_EN] is set, the controller begins generating address parity while monitoring the error signal sent back from the registered DIMMs.

However, the controller may detect a false address parity error when the memory controller is first enabled, because the register could be in a state where it is driving the parity error signal active low.

Impact: ERR_DETECT[APE] are erroneously set by hardware, and an interrupt is propagated if ERR_INT_EN[APEE] is set.

Workaround: Before setting DDR_SDRAM_CFG[MEM_EN], set ERR_DISABLE[APED]. After setting DDR_SDRAM_CFG[MEM_EN], issue a read back to DDR_SDRAM_CFG. Then, clear ERR_DISABLE[APED] if error reporting is desired for address parity.

Fix plan: No plans to fix

DDR 2: The memory controller does not use the correct default impedances for the IOs in DDR3 mode

Description: If the memory controller is operating in full-strength mode without software overrides or hardware calibration via DDRCDR_1/DDRCDR_2, then it uses a 40-Ω target for the IO drivers. If the memory controller is operating in half-strength mode (DDR_SDRAM_CFG[HSE] = 1), then the default values used for the IOs have a much higher impedance. The full strength default should have targeted 19–20 Ω, and the half-strength default should have targeted about 40 Ω. In addition, the DDR controller does not allow driver calibration to train with an 18–20 Ω precision resistor. Instead, it only supports training in DDR3 mode with higher impedances (~ 40 Ω).

Impact: If incorrect driver impedances are used, then setup/hold violations may be present on the DDR interface. The inability to calibrate to an 18-Ω value is not considered a large impact, as it is expected that customers train to the impedance desired for the data (about 40 Ω).

Workaround: If training will not be used, then the following values should be programmed into the software overrides for each group in the DDRCDR_1/DDRCDR_2 registers:

- If half-strength (~ 40 Ω) is desired for a particular group, set the DSO_*PZ and DSO_*NZ fields to 4'b0111 (also set DSO_*_EN).
- If full-strength (~ 19–20 Ω) is desired for a particular group. set the DSO_*PZ and DSO_*NZ fields to 4'b1010 (also set DSO_*_EN).

Note that leaving DDR_SDRAM_CFG[HSE] cleared without any overrides sets all groups to the 40-Ω targeted setting. However, this may be targeted to a much lower impedance setting (~20 Ohms) in future revisions. If training is used for the half-strength values, then the full-strength groups can still have software overrides as described above.

Fix plan: No plans to fix

DDR 3: MCKE signal may not function correctly at assertion of $\overline{\text{HRESET}}$

Description: During the assertion of $\overline{\text{HRESET}}$ (excluding the initial power-on-reset) the device may erroneously drive the state of MCKE to the incorrect level or release it to high impedance after removing the clocks from the DRAM. This could place the DRAMs into an undefined state causing future operations to fail. The primary fail mechanism is for the device to incorrectly train its I/O receivers during DDR initialization.

There are power on reset configuration signals sampled during $\overline{\text{HRESET}}$ that do not quickly achieve correct values using their internal pull-ups. As a result, the device may be temporarily placed into a test mode.

The POR configuration pin used to enter test mode is MSRCID[2]. If MSRCID[2] is driving a zero at the time of $\overline{\text{HRESET}}$ assertion, then MCKE[0:1] may drive a one.

The DDR controller should drive the MCKE[0:1] pins active low throughout and after $\overline{\text{HRESET}}$ assertion. However, when the DDR controller enters a test mode, the DDR MCKE driver is released to high impedance or driven high, preventing the MCKE[0:1] pins from being driven low immediately after $\overline{\text{HRESET}}$ assertion.

Note that when the controller is configured for DDR3 mode, the DRAMs have a $\overline{\text{RESET}}$ pin that can be controlled by the $\overline{\text{HRESET}}$ pin on our device to reset the DRAMs everytime an $\overline{\text{HRESET}}$ is asserted on our device. This eliminates the JEDEC requirement for MCKE to be driven low at $\overline{\text{HRESET}}$ assertion.

Impact: The DRAMs may erroneously enter an undefined state preventing the completion of read operations during DRAM initialization sequence. This may result in an auto calibration error (ERR_DETECT[ACE]) or improper training during the initialization sequence. A failure to train properly may result in corrupted data transfers to and from DDR.

Workaround: There are several possible workarounds. Depending on the application, select one of the following options:

Option 1

Do not use chip select 0 or 1 on DDR controller #1.

Option 2

At assertion of $\overline{\text{HRESET}}$ perform an alternative DDR controller initialization sequence for each utilized controller. This clears the DRAM state machines and allows them to operate properly. Before this sequence is implemented do not enable any DDR LAWBAR entries. Details of alternative sequence are as follows:

- D2 offset is CCSRBAR + DDR_OFFSET + 0xf04
- D3 offset is CCSRBAR + DDR_OFFSET + 0xf08

1. Configure DDR registers as is done in normal DDR configuration. Do not set DDR_SDRAM_CFG[MEM_EN].
2. Set reserved bit EEBACR[3] at offset 0x1000.
3. Before DDR_SDRAM_CFG[MEM_EN] is set, write DDR_SDRAM_CFG_2[D_INIT].
4. Before DDR_SDRAM_CFG[MEM_EN] is set, write D3[21] to disable data training.
5. Wait 200 µs (as described in the section “DDR SDRAM Initialization Sequence,” in the applicable device reference manual)
6. Set DDR_SDRAM_CFG[MEM_EN].
7. Poll DDR_SDRAM_CFG_2[D_INIT] until it is cleared by hardware.
8. Clear D3[21] to re-enable training.

9. Set D2[21] to force the data training to run.
10. Poll on D2[21] until it is cleared by hardware.

    After this step there are two options that can be followed if ECC is enabled before continuing on to step 11 . If DDR ECC is not utilized enable the DDR LAWBARs and continue to step 11 . Sub-Option 1 requires a calculated delay. Sub-Option 2 does not require the delay, but it is not supported for applications with DDR interleaving enabled.

    Sub-Option 1

    a. Wait calculated delay

       Required delay for 64-bit DDR2 can be calculated as follows:

       Delay = 400 ms/Gbytes × max memory size

       For 32-bit data buses, multiply this number by 2.

       Example: assume 64-bit DDR2, memory size = 1 Gbyte

       Delay = 400ms/Gbytes × 1 Gbyte = 400 ms
    b. Set DDR_SDRAM_CFG_2[D_INIT]
    c. Poll on DDR_SDRAM_CFG_2[D_INIT] until it is cleared by hardware, then the system can proceed.
    d. Enable any DDR LAWBAR entries and proceed to step 11 .

    Sub-Option 2

    a. Enable any DDR LAWBAR entries.
    b. Set ERR_DISABLE[MBED] and ERR_DISABLE[SBED] to disable SBE and MBE detection.
    c. Complete a 32-byte non-snoopable DMA transaction with the source and destination address equal to the DDR initialization address which is either the starting address of CS0_BNDS by default or programmed in DDR_INIT_ADDR.
    d. After the DMA transaction has completed clear ERR_DISABLE[MBED] and ERR_DISABLE[SBED] to enable SBE and MBE detection as desired for specific applications.

11. Clear reserved bit EEBACR[3] at offset 0x1000.

Option 3

Use an active component (for example, CPLD) to drive MCKE signals to the DRAMs. Inputs to this logic should include MCKE and $\overline{\text{HRESET\_REQ}}$, both from the device. When $\overline{\text{HRESET\_REQ}}$ asserts, the MCKE signal to the DRAMs should be driven low by the active component. When $\overline{\text{HRESET\_REQ}}$ is negated, the MCKE value driven by the CPLD should match the value driven by the device. The JEDEC defined t_{Delay} parameter between the MCKE and MCK/$\overline{\text{MCK}}$ signals must also be controlled by this workaround. In addition, note that MCKE must still meet all JEDEC-defined ADDR/CMD setup/hold requirements when using the external component to help drive MCKE.

Option 4

Power cycle the DRAM during $\overline{\text{HRESET}}$ assertions.

Fix plan: No plans to fix

DDR 4: DDR controller may not work with dual-ranked DDR3 registered DIMM if write leveling is enabled

Description: According to JEDEC specifications, the DDR controller must allow 3 DRAM cycles between back-to-back MRS (mode register set) commands when using DDR3 registered DIMMs. However, during write leveling, the DDR controller currently sends MRS commands on consecutive cycles (or on every other cycle for 2T timing mode). Therefore, the 3 DRAM cycles required between back-to-back MRS commands is not met in 1T or 2T timing mode.

Impact: DDR3 registered DIMMs with the write leveling may fail or yield invalid results. This could cause corrupt data to be written to DRAM during future write accesses.

Workaround: There are several possible workarounds. Depending on the application select one of the following options:

- Use unbuffered DDR3 DIMMs.
- Use single-ranked DDR3 DIMMs.
- Disable write leveling if dual-ranked DDR3 registered DIMMs are used.
- Software Workaround #1 (for use with dual-ranked DDR3 registered DIMMs with write leveling enabled):

  Note the following DEBUG register:

  DEBUG_2 offset is CCSRBAR + DDR_OFFSET + 0xF04

  Set DDR_SDRAM_CFG[3T_EN] when configuring DDR registers as is done in normal configuration before DDR_SDRAM_CFG[MEM_EN] is set.

  Note: If it is preferred to use 1T timing for better performance, then DDR_SDRAM_CFG[3T_EN] may be cleared via the following sequence after the DDR controller has completed the initialization sequence (If DDR_SDRAM_CFG_2[D_INIT] was set prior to DDR_SDRAM_CFG[MEM_EN], D_INIT should be polled until it is cleared by hardware).

  a. Set DDR_SDRAM_CFG[MEM_HALT].
  b. Poll on DEBUG_2[30] until it is set by hardware.
  c. Set DDR_SDRAM_INTERVAL[REFINT] = 4'h0000 to Disable refreshes.
  d. Clear DDR_SDRAM_CFG[3T_EN].
  e. Set DDR_SDRAM_INTERVAL[REFINT] back to the original value.

     If ECC is not enabled, go to f below. If ECC is enabled, then re-initialize the DRAM and go to the step below:

     1. Set DDR_SDRAM_CFG_2[D_INIT] .
     2. Poll on DDR_SDRAM_CFG_2[D_INIT] until it is cleared by hardware.
     3. Wait calculated delay.

        Required delay for 64-bit DDR2 can be calculated as follows:

        Delay = 400 ms/Gbytes × size of memory being initialized on the controller.

        For 32-bit data buses, multiply this number by 2.

        Example: assume 64-bit DDR2, memory size = 1 Gbyte.

        Delay = 400 ms/GByte × 1 Gbyte = 400 ms

  f. Clear DDR_SDRAM_CFG[MEM_HALT].

Fix plan: No plans to fix

DDR 5: DDR to CCB clock ratios of greater than 1.5:1 can result in DDR configuration register corruption

Description: If the clock ratio between the DDR and CCB is greater than 1.5:1, the DDR memory-mapped registers may be corrupted during writes and the memory controller may return incorrect data during reads.

Impact: DDR controller fails to function correctly.

Workaround: The DDR to CCB clock ratio must be kept at or below 1.5:1. For example, if a DDR data rate of 600 MHz is used, the minimum CCB frequency should be 400 MHz.

Fix plan: Fixed in Rev. 1.1

DDR 6: Asserting panic interrupt via $\overline{\text{IRQ\_OUT}}$ the DDR controller may violate the JEDEC specifications

Description: The DDR controller supports a panic interrupt that can be controlled by hardware. This interrupt can be used to place the DRAMs into self-refresh mode. However, the DDR controller can enter a bad state and violate JEDEC specifications for entering self-refresh when this interrupt is used.

Impact: DRAM contents may become corrupted when self-refresh is entered. There may be other impacts related to the DRAM devices when the JEDEC specs are violated.

Workaround: Instead of using the panic interrupt, the DDR controller can be placed into self-refresh without waiting on the part to enter a sleep state by asserting DDR_SDRAM_CFG_2[FRC_SR].

Fix plan: No plans to fix

DDR 7: The DDR controller may fail the write leveling sequence for DDR3

Description: During write leveling, JEDEC allows the DDR3 SDRAM to drive status back to the DDR controller on either all DQ bits or the prime DQ bit. The prime DQ bit is typically defined as DQ[0] of each byte. The DDR controller currently uses DQ[0,8,16,24,32,40,48,56] and ECC[0] to read the status back for the various bytes of data. However, some DIMMs will be routed such that these data bits will not be connected to DQ[0] of each discrete device. Hence, if a particular vendor only drives status on a single bit during write leveling, this bit may get routed to a DQ bit that is different than what the DDR controller is expecting. Then, the controller will not observe the status correctly and will fail the write leveling sequence for DDR3.

Impact: The write leveling sequence may fail, which could prevent the DDR controller from writing correctly to DRAM. This does not affect a system using DRAM discrete devices. It only affects a system using DDR3 DIMMs that drive the write leveling response on different data line than controller is expecting (that is, many DDR3 DIMM from various vendors will work, but others may not).

Workaround: Use one of the following options.

- Use DDR3 memories that drive write leveling status on all DQ bits instead of a single DQ bit.
- Modify the board design to connect DQ[0,8,16,24,32,40,48,56] and ECC[0] of the part to the bits carrying status on the DIMM connector. Depending upon the raw card design used, this may be different. This workaround limits the DIMM topologies that can be used for a particular system. For example, a single-ranked DIMM may have different routing for DQ[0] from each discrete than a dual-ranked DIMM.
- Disable write leveling for DDR3 via DDR_SDRAM_WRLVL_CNTL[WRLVL_EN]. Instead, TIMING_CFG_2[WR_DATA_DLY] can be used to align DQS with MCK at the DRAM.

Fix plan: No plans to fix

DDR 8: Data may become corrupted for DDR3 32-bit bus mode

Description:	When the DDR controller is operating in 32-bit bus mode using 8-beat bursts, it may corrupt the data during 32-byte accesses. The controller may incorrectly send the wrong data beats.
Impact:	No ECC error will be detected, as the error occurs past the ECC checking, and a full doubleword will be incorrect. Therefore, the incorrect data will be sent, and undefined results could occur. This only affects DDR3 mode.
Workaround:	Write register at CCSRBAR Offset 0xE_0F34 with a value of 0x4000 0000, to allow DDR3 at 32-bit bus mode to operate properly.
Fix plan:	No plans to fix

DMA-A001: Uncorrectable data read errors for DMA transfers may also cause DMA lockup or additional detectable data corruption

Description: If an uncorrectable ECC error, system bus timeout, or illegal address occurs in the system, an additional unexpected impact can cause occasional corruption of unrelated DMA streams (or, if multiple such errors occur, DMA lockup). Generally, it is expected that the measures used to deal with the initial read error mitigate any impact of this unexpected behavior.

DMA lockup is only realistic for programming errors and hard errors such as bad memories because it is not expected that the system will continue to operate without reset in the face of repeated uncorrectable read errors or timeouts, etc. Forwarding of corrupt data due to individual soft errors can be avoided as described in workaround option 2 below.

Freescale discovered this issue while debugging a test case in which intentionally injected programming errors caused a stream of illegal addresses that eventually resulted in a DMA controller hang.

The DMA controller can process up to 16 total read and write transactions at a time from up to 4 active channels. Under normal conditions, the reads return data which is then used to program the controller (descriptor read) or forwarded to the corresponding write address(es) (data read). If an error occurs on a read transaction, however, the controller may mis-handle the response, and end up corrupting the queue state in the DMA. This in turn may corrupt in-flight data streams (detectable). If multiple error responses occur, the DMA may lock up (channel stays busy forever and does not halt) and be unable to process new transactions.

Errors on reads can be from several sources, including the following:

- Illegal address (does not match valid LAW or CCSRBAR)
- Illegal target settings (for example, address maps to PCI-Express, but ATMU settings for that address are invalid)
- Target error (for example, multi-bit ECC error in DDR memory)
- End-to-end error (for example, error response packet on PCI-Express or Serial RapidIO)

Impact: The impact of this behavior is expected to be low in the context of the hard system failure or data corruption or programming error that caused the behavior. That is, the measures taken to deal with the initial problem, such as taking the system offline (for system repair or software patch) or system reset (for soft errors) also take care of the DMA lockup or corrupted DMA transfer.

Below is a description of the possible additional impact of a failed DMA read.

An error on a DMA read transaction may corrupt the programmed state or data for a subsequent transfer from the same or a different channel.

A sequence of errors may cause DMA lockup in which the DMA is unable to process new transactions. In this case, at least one channel stays busy forever and never halts.

In all cases, SRn[TE] is set for channel n that originated the read transaction that had the error.

For DMA lockup, at least one channel, not necessarily the one that has SRn[TE] set, stays busy forever and does not halt.

Not every error causes or contributes to DMA corruption or lockup. A minimum of 4 errors is necessary to cause DMA lockup. A single error can cause data corruption if the DMA issues another transaction within a small window after an error response arrives for a prior transaction. Alignment and transfer size affect the possibility of new transactions falling in this window, as described in workarounds below.

Neither DMA lockup nor data corruption prevents software from being notified of the error condition via interrupt (nor does this erratum limit the ability of software to query error reporting registers or status registers).

Workaround: The following steps can be taken to avoid any of the unexpected behavior caused by a DMA read data failure in case the steps taken to resolve the original data failure are not adequate to mitigate the additional unexpected behavior:

- Option 1: Limit DMA operation to 1 channel per DMA controller, and limit each descriptor to:
  - Byte count to ≤ 1 Kbyte. Byte count is set in BCRn[BC].
  - Aligned start addresses, such that each descriptor yields no more than 8 transfers (reads and writes, for example, 256 bytes aligned for a 1-Kbyte transfer yields 4 reads and 4 writes). Start addresses for reads are set in SADRn, and for writes in DADRn.
  - Use write-with-response for last write of descriptor (DATRn[NLWR]=0).

- Option 2: Limit DMA operation to 1 channel per DMA controller, and limit writes to 64 bytes aligned start addresses. In this scenario, DMA lockup is still possible, but not data corruption.

- Option 3: Ensure DMA reads do not get an error response by taking the following actions:
  a. Validate that all DMA descriptor, source and destination addresses reference only valid LAW or CCSRBAR addresses, and valid target chip selects.
  b. Do not host DMA descriptors on, or use the DMA engine to read from, external interfaces (for example, PCI-Express, serial RapidIO), which can return error responses. Writes to external interfaces are acceptable.
  c. Disable ECC checking in the DDR controller by setting DDR_SDRAM_CFG[ECC_EN] = 0.
  d. If hosting DMA descriptors on the eLBC, or using the DMA engine to read from the eLBC, then disable eLBC parity/ECC and timeout error checking by setting each of LTEDR[BMD, FCTD, PARD] = 1.
  e. If L2 cache is enabled, or if hosting descriptors in L2 SRAM or reading from L2 as SRAM, then disable L2 parity/ECC checking by setting each of L2ERRDIS[TPARDIS, MBECCDIS] = 1.

- Option 3a: If detection of multi-bit ECC errors is not required in the system, then same as option 3 except for step c, which is replaced by the following:

  c. Set ERR_DISABLE[MBED] = 1 and ERR_SBE[SBET] = 0xFF, and periodically poll and clear ERR_SBE[SBEC] to ensure it does not reach 0xFF.

NOTE

To switch from option 3 or 3a to option 1 or 2, in order to use the DMA controller to read from interfaces which may generate errors, software may halt the other DMA channels by setting MRn[CS] = 0 and wait for SRn[CH] = 1, then program the idle channel as in option 1 or option 2. When the transaction reading from the error-susceptible interface is complete (MRn[CB] = 0), resume multi-channel operation as in option 3 or 3a by setting MRn[CS] = 1 for the other 3 halted channels.

Fix plan: No plans to fix

DUART 1: BREAK detection triggered multiple times for a single break assertion

Description: A UART break signal is defined as a logic zero being present on the UART data pin for a time longer than (START bit + Data bits + Parity bit + Stop bits). The break signal persists until the data signal rises to a logic one.

A received break is detected by reading the ULSR and checking for BI = 1. This read to ULSR clears the BI bit. After the break is detected, the normal handling of the break condition is to read the URBR to clear the ULSR[DR] bit. The expected behavior is that the ULSR[BI] and ULSR[DR] bits do not get set again for the duration of the break signal assertion. However, the ULSR[BI] and ULSR[DR] bits continue to get set each character period after they are cleared. This continues for the entire duration of the break signal.

At the end of the break signal, a random character may be falsely detected and received in the URBR, with the ULSR[DR] being set.

Impact: The ULSR[BI] and ULSR[DR] bits get set multiple times, approximately once every character period, for a single break signal. A random character may be mistakenly received at the end of the break.

Workaround: The break is first detected when ULSR is read and ULSR[BI]=1. To prevent the problem from occurring, perform the following sequence when a break is detected:

1. Read URBR, which returns a value of zero, and clears the ULSR[DR] bit
2. Delay at least 1 character period
3. Read URBR again, which return a value of zero, and clears the ULSR[DR] bit

ULSR[BI] remains asserted for the duration of the break. The UART block does not trigger any additional interrupts for the duration of the break.

This workaround requires that the break signal be at least 2 character-lengths in duration.

This work around applies to both polling and interrupt-driven implementations.

Fix plan: No plans to fix

eLBC1: Multi-bank DRAM and SDRAM do not work without any external logic

Description: When the UPM is connected to multi-bank DRAM and SDRAM, the memories require that the bank address should be driven during both RAS and CAS cycles. However, due to the internal implementation, this does not happen.

Impact: Multi-bank memories (DRAM/SDRAM) do not work as expected with the UPM.

Workaround: An external intelligent logic can be used on the board as a workaround.

Fix plan: No plans to fix

eLBC 2: LTEATR and LTEAR may show incorrect values under certain scenarios

Description: When FCM special operation is in progress, if any one of the errors/events that are listed in LTESR (except chip select error) occurs, the address and attribute that are captured in LTEAR and LTEATR may be incorrect.

Impact: LTEAR and LTEATR cannot be used for debugging in this scenario.

Workaround: None

Fix plan: No plans to fix

eLBC-A001: Simultaneous FCM and GPCM or UPM operation may erroneously trigger bus monitor timeout

Description: When the FCM is in the middle of a long transaction, such as NAND erase or write, another transaction on the GPCM or UPM triggers the bus monitor to start immediately for the GPCM or UPM, even though the GPCM or UPM is still waiting for the FCM to finish and has not yet started its transaction. If the bus monitor timeout value is not programmed for a sufficiently large value, the local bus monitor may time out. This timeout corrupts the current NAND Flash operation and terminate the GPCM or UPM operation.

Impact: Local bus monitor may time out unexpectedly and corrupt the NAND transaction.

Workaround: Set the local bus monitor timeout value to the maximum by setting LBCR[BMT] = 0 and LBCR[BMTPS] = 0xF.

Fix plan: No plans to fix

eSPI-A001: Master out slave in (MOSI) master data output hold time tNIKHOX should be 3.3 ns for temperatures less than 0° C

Description: The MOSI master data output hold time tNIKHOX in the full cycle mode for extended low temperature products was defined as 4.0 ns minimum. However, it should be 3.3 ns minimum. The parameter is only affected for the extended low temperature when the SPCOM[RxDelay] bit is set.

Impact: Interfacing to a device that requires an output hold time greater than 3.3 ns is not possible at an extended low temperature for the full cycle mode.

Workaround: Use products that require a hold time of 3.3 ns or less, or use SPCOM[RxDelay]=0, or guarantee that the system does not experience extended low temperatures outside the standard temperature range.

Fix plan: No plans to fix

eTSEC 1: Frame is dropped with collision and HALFDUP[Excess Defer] = 0

Description: eTSEC drops excessively deferred frames without reporting error status when HALFDUP[Excess Defer] = 0. This erratum affects 10/100 Half Duplex modes only.

Impact: The eTSEC does not correctly abort frames that are excessively deferred. Instead it closes the BD as if the frame is transmitted successfully. This results in the frame being dropped (because it is never transmitted) without any error status being reported to software.

Workaround: Do not change HALFDUP[Excess Defer] from its default of 1.Thus eTSEC always tries to transmit frames regardless of the length of time the transmitter defers to carrier.

Fix plan: No plans to fix

eTSEC 2: Fetches with errors not flagged, may cause livelock or false halt

Description: The error management for address (for example, unmapped address) and data (for example, multi-bit ECC) errors in the Ethernet controller does not properly handle all scenarios. The behavior is as follows:

Scenario 1: Address error on Tx data fetch

The Ethernet controller does not detect errors on Tx data fetches. IEVENT[EBERR] is not set and the queues are not halted. For address errors, no data is returned to eTSEC and the controller hangs. The error may still be detected at the platform level, via an interrupt from the source of the error (e.g. ECM/MCM address mapping error).

Scenario 2: Data error on Tx data fetch

The Ethernet controller does not detect errors on Tx data fetches. IEVENT[EBERR] is not set and the queues are not halted. For data errors, the frame is transmitted as if data is good, with good FCS. The error may still be detected at the platform level, via an interrupt from the source of the error (e.g. DDRC multibit ECC error).

Some fetch errors are handled correctly. The correct behavior is as follows:

1. Non-first TxBD fetch for queue 0 OR TxBD fetch for queues 1-7

   The ethernet controller will set IEVENT[EBERR] and halt all Tx queues (TSTAT[THLTn]=1, n=0-7). EDIS[EBERRDIS] must be 0.

2. RxBD fetch

   The ethernet controller will set IEVENT[eberr] and halt the queue with the error (RSTAT[QHLTn]=1). EDIS[EBERRDIS] must be 0.

Impact: The Ethernet controller may stop transmitting packets without setting IEVENT[EBERR] if a buffer descriptor or data fetch has an uncorrectable error.

The transmit scheduler may halt queues without setting IEVENT[EBERR] if a buffer descriptor fetch has an uncorrectable error.

The controller does not detect errors on Tx data fetches and transmits corrupted data without an error indicator.

Workaround: All scenarios:

1. Make sure all eTSEC BD and data addresses map to valid regions of memory.
2. Ensure EDIS[EBERRDIS] = 0.

For recovery from error scenarios 1:

The Tx reset sequence is:

1. Set DMACTRL[GTS]
2. Poll IEVENT[GTSC] until set or 10,000 byte times elapse
3. Clear MACCFG1[Tx_Flow]
4. Wait 256 TX_CLK cycles
5. Clear MACCFG1[Tx_EN]
6. Wait 3 TX clocks
7. Set MACCFG1[Reset Tx MC] and MACCFG1[Reset Tx fun]
8. Wait 3 TX clocks
9. Clear MACCFG1[Reset Tx MC] and MACCFG1[Reset Tx fun]
10. Set TBPTRn to next available BD in the TX ring.

11. Set MACCFG1[Tx EN] and, if desired, MACCFG1[Tx Flow]
12. Clear DMACTRL[GTSC]

For Scenario 2, data errors can be flagged by platform for additional software processing, but no workaround exists to prevent the transmission of corrupted data.

Fix plan: No plans to fix

eTSEC 3: Multiple BD frame may cause hang

Description: Software must expect eTSEC to prefetch multiple TxBDs, and for TCP/IP checksumming an entire frame must be read from memory before a checksum can be computed. Accordingly, the R bit of the first TxBD in a frame must not be set until at least one entire frame can be fetched from this TxBD onwards. If eTSEC prefetches TxBDs and fails to reach a last TxBD (with bit L set), it halts further transmission from the current TxBD ring and report an underrun error as IEVENT[XFUN]; this indicates that an incomplete frame was fetched, but remained unprocessed.

If software sets up a frame with multiple BDs, and sets the first BD READY bit before the remaining BDs are marked ready, and if the controller happens to prefetch the BDs when some are marked ready and some marked unready, the controller may not halt or set IEVENT[XFUN], hanging the transmit.

Impact: If software does not follow the guidelines for setting the ready bit of the first BD of a multiple TxBD frame, the Ethernet controller may hang.

Workaround: Software must ensure that the ready bit of the first BD in a multiple TxBD frame is not set until after the remaining BDs of the frame are set ready.

Fix plan: No plans to fix

eTSEC 4: VLAN Insertion corrupts frame if user-defined Tx preamble enabled

Description: When TCTRL[VLINS] = 1, the VLAN is supposed to be inserted into the Tx frame 12 bytes after start of DA (after DA and SA). If user-defined Tx preamble is enabled (MACCFG2[PreAmTxEn] = 1), the VLAN ID is inserted 12 bytes after the start of the preamble (4 bytes after start of DA), thus overwriting part of DA and SA.

Impact: If VLAN insertion is enabled with user-defined Tx preamble, the VLAN ID corrupts the Tx frame destination and source addresses.

Workaround: Use one of the following workarounds:

- Disable user-defined Tx preamble by setting MACCFG2[PreAmTxEn] = 0.
- Disable VLAN insertion by setting TCTRL[VLINS] = 0.

Fix plan: No plans to fix

eTSEC 5: User-defined Tx preamble incompatible with Tx Checksum

Description: If user-defined Tx preamble is enabled (by setting MACCFG2[PreAmTxEn]=1), an extra 8 bytes of data is added to the frame in the Tx data FIFO. IP and TCP/UDP checksum generation do not take these extra bytes into account and write to the wrong locations in the frame.

Impact: Enabling both user-defined Tx preamble and IP or TCP/UDP checksum causes corruption of part of the corresponding header.

Workaround: Use one of the following workarounds:

- Disable user-defined Tx preamble by setting MACCFG2[PreAmTxEn] = 0.
- Disable IP and TCP/UDP checksum generation by setting TCTRL[IPCSEN]=0 and TCTRL[TUCSEN] = 0.

Fix plan: No plans to fix

eTSEC 6: Transmit fails to utilize 100% of line bandwidth

Description: The minimum interpacket gap (IPG) between back-to-back frames is 96 bit times. To ensure 100% utilization of an interface, the maximum gap between back-to-back streaming frames should also be 96 bit times (12 cycles). The Tx portion of the Ethernet controller may fail to meet that requirement, depending on mode, clock ratio, and internal resource contention.

- For single-queue operation, IPG is always 12 cycles.
- For multiple queue operation with fixed priority scheduling, IPG for back-to-back frames from different queues varies between 70–140 cycles.
- For multiple queue operation with round-robin scheduling, IPG for back-to-back frames from different queues is on the order of 20–40 cycles longer than multiple queue operation with fixed priority.

In all cases, the impact of longer IPG is greater for smaller frames. With multiple queue operation, small frames may also increase the gap, as the buffer descriptor (BD) prefetching may fall behind the data rate, especially at lower clock ratios.

Impact: Tx bandwidth cannot achieve 100% line rate, especially for multiple queue operation or relatively small frames.

Workaround: The following options maximize the bandwidth utilized by the Ethernet controller.

- If multiple Tx queue operation is not required, use single Tx queue operation (thus eliminating the extra gap caused by switching queues) and use frames larger than 64 bytes (thus reducing the IPG as a portion of total bandwidth).
- If multiple Tx queue operation is required, use priority arbitration by setting TCTRL[TXSCHED]=2'b01 and maximize the number of BDs enabled per ring to minimize switching between rings. Also, minimize use of small frames, thus reducing IPG as a portion of total bandwidth.

Fix plan: No plans to fix

eTSEC 7: Unexpected babbling receive error in FIFO modes

Description: In MAC modes, a babbling receive error occurs if MACCFG2[Huge Frame]=1 and a receive frame exceeds MAXFRM. There is no MAXFRM in FIFO modes, so there should be no babbling receive errors. The Ethernet controller does not qualify the babbling receive error with interface mode, so a babbling receive error will be reported if a receive frame on a FIFO interface exceeds the value of MAXFRM.

Impact: The controller may erroneously report babbling receive errors in FIFO mode.

Workaround: In FIFO modes, disable interrupts for babbling receive errors by setting IMASK[BREN]=0, and ignore any setting of IEVENT[BABR].

Fix plan: No plans to fix

eTSEC 8: ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software

Description: The MIB function of the Ethernet controller has a feature to automatically zero out the registers when reading them if ECNTRL[AUTOZ] = 1. If the register read occurs in the same cycle as a hardware update of the register, then the register clear will not occur. Any software periodically reading MIB registers would expect to read A the first time, B the second, and C the third, with each value representing only the events that occurred in the interval between reads. If the first read collides with a hardware update, the second read would return A + B instead of B.

Hardware updates for MIB registers occur once per frame. For streaming 64-byte frames, the update would be every 84 Rx or Tx clocks (8 bytes of preamble, 64 bytes of data and 12 cycles of IPG).

Impact: Software polling of MIB counters with ECNTRL[AUTOZ] = 1 will over an extended period read a larger number of events than actually seen by the controller.

Workaround: Disable automatic clearing of the MIB counters by writing ECNTRL[AUTOZ] = 0. Software routines which periodically read MIB counters and accumulate the results should accumulate only when an MIB counter overflows, as in the description that follows: Assuming a 32-bit MIB counter (MIB_VALUE), a 64-bit accumulator consisting of two 32-bit registers (ACCUM_HI, ACCUM_LO), and a Carry Out bit (ACCUM_LO_CO), change the 64-bit accumulator update as follows:

Previous accumulate method (with ECNTRL[AUTOZ] = 1):

```
// Accumulate the MIB_VALUE into the lower half of the accumulator
 {ACCUM_LO_CO,ACCUM_LO} = {1'b0,ACCUM_LO} + {1'b0,MIB_VALUE};
// Accumulate the Carry Out from the step above, as well as the MIB register
OVFRFLW, which is detected through the CARn register.
{ACCUM_HI_CO,ACCUM_HI} = {1'b0,ACCUM_HI} + ACCUM_LO_CO + OVRFLW;
```

New accumulate method (with ECNTRL[AUTOZ]=0):

```
// Read instead of accumulate since we are not clearing MIB_VALUE
 ACCUM_LO = MIB_VALUE;
// Accumulate the MIB register OVRFLW, which is detected through the CARn
register
 {ACCUM_HI_CO,ACCUM_HI} = {1'b0,ACCUM_HI} + OVRFLW;
```

Fix plan: No plans to fix

eTSEC 9: Controller stops transmitting pause control frames

Description: There are three types of events that trigger a transmit pause control frame:

1. Software sets TCTRL[TFC_PAUSE]=1
2. The Rx data FIFO exceeds its predetermined threshold
3. RCTRL[LFC]=1 and the number of free BDs falls below the programmed threshold.

If a second pause request event (of the same or different type) occurs near the end of the transmission a pause control frame, the pause state machine may not detect that the transmission of the first pause is complete, and will therefore fail to start transmitting the second or any subsequent requested pause control frame. Only a Tx state machine reset will restore the ability to transmit pause control frames.

Note that for the purposes of determining the likelihood of a second pause frame request occuring at the end of transmitting a previous pause frame, the time it takes to transmit the first pause control frame includes waiting for any data frame already in progress to complete transmission. For jumbo or huge frames, that is (9608 + 20 + 72) × 8 = 77,600 bit times. For MAXFRM=1536 frames, that is (1544+20+72) × 8 = 13,088 bit times.

Impact: If two pause control triggering events occur within the affected window of time, the controller will stop transmitting pause control frames. As a result, the external system may continue transmitting frames to the device even though there is a condition which requires a pause in receiving frames (Rx FIFO almost full, free Rx BDs below threshold, software request). The extra received frames may be dropped if there is a BSY condition (no Rx BDs available) or if the Rx FIFO is full.

Workaround: To prevent the error condition, set MACCFG1[Tx_Flow] = 0, thus disabling all three sources of pause frame generation.

Options to minimize the frequency of the error condition:

The frequency of hitting the error condition is directly proportional to the frequency of pause frame generation. Following one or more of the following options will reduce the frequency of pause frame generation:

1. Minimize or avoid the use of software-generated pause control frames.

   Pause control frames are generated by software by setting TCTRL[TFC_PAUSE] = 1. Note that if the pause control state machine is stuck, neither IEVENT[GTSC] nor IEVENT[TXC] will be set to 1 as a result of setting TCTRL[TFC_PAUSE] = 1, and TFC_PAUSE will never be reset to 0. Transmission of data frames will continue while TFC_PAUSE stays stuck at 1.

2. Do not use lossless flow control. Lossless flow control is enabled by setting RCTRL[LFC] = 1.
3. If using lossless flow control, increase the value of PTV such that two LFC-triggered pause frames cannot occur within the failing window.

   For a system supporting jumbo or huge frames, a PTV setting of 80 pause quanta (40,960 bit times) is sufficient to prevent back-to-back LFC-triggered pause frames. For a system not supporting jumbo or huge frames, with MAXFRM set to 1536, a PTV setting of 15 pause quanta (7680 bit times) is sufficient.

4. Limit the number of pause frames generated due to data in RxFIFO exceeding the threshold by setting the eTSEC register at offset 0x050 to 0. By doing so, pause frames due to RxFIFO threshold will only be generated if the RxFIFO overflows.

Detection and recovery:

This detection and recovery mechanism requires that flow control and lossless flow control are both enabled (MACCFG1[Tx_Flow] = 1 and RCTRL[LFC] = 1).

Program the RxFIFO threshold as in item 4 of the "Options to minimize the frequency of the error condition" list and use the following detection mechanism:

1. Enable BSY interrupts by setting IMASK[BSYEN] = 1 and EDIS[BSYDIS] = 0.
2. On receiving a BSY interrupt, write a 1 to IEVENT[BSY] to acknowledge the event
3. Assume the pause state machine is stuck and proceed with the following recovery mechanism:
    a. Set DMACTRL[GTS] = 1 to perform a graceful transmit stop
    b. Wait for IEVENT[GTSC] to be set, indicating stop complete
    c. Write a 1 to IEVENT[GTSC] to acknowledge the event
    d. Toggle MACCFG1[Tx_En] 1->0->1
    e. Set DMACTRL[GTS]=0 to clear the graceful transmit stop

Fix plan: Fixed in Rev 1.1

eTSEC 10: Half-duplex collision on FCS of Short Frame may cause Tx lockup

Description: In half-duplex mode, if a collision occurs in the FCS bytes of a short (fewer than 64 bytes) frame, then the Ethernet MAC may lock up and stop transmitting data or control frames. The problem can only occur if MACCFG2[PAD/CRC] = 0 and MACCFG2[CRCEN] = 1. Only a reset of the controller can restore proper operation once it is locked up.

Impact: A collision on hardware-generated FCS bytes of a short frame in half-duplex mode may cause a Tx lockup.

Workaround: **Option 1:**

Set MACCFG2[PAD/CRC] = 1, which pads all short Tx frames to 64 bytes.

Option 2:

Use software-generated CRC (MACCFG2[PAD/CRC] = 0, MACCFG2[CRC EN] = 0 and TxBD[TC] = 0)

Fix plan: No plans to fix

eTSEC 11: Magic Packet Sequence Embedded in Partial Sequence Not Recognized

Description: The Ethernet MAC should recognize Magic Packet sequences as follows:

Any Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of data payload. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFs, followed by 16 copies of the MAC's unique IEEE station address in the normal byte order for Ethernet addresses.

If a complete Magic Packet sequence (including 6 bytes of 0xFF) immediately follows a partial Magic Packet sequence, however, the complete sequence will not be recognized and the MAC will not exit Magic Packet mode.

The following are example partial sequences followed by the start of a complete sequence for station address 01_02_03_04_05_06:

- Sequence a) FF_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

  Seventh byte of 0xFF does not match next expected byte of Magic Packet Sequence (01). Pattern search restarts looking for 6 bytes of FF at byte 01.

- Sequence b)
  FF_FF_FF_FF_FF_FF_01_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

  First FF byte following 01 does not match Magic Packet sequence.

  Pattern search restarts looking for 6 bytes of FF at second byte of FF following 01.

The following is an example partial sequence followed by the start of a complete sequence that is erroneously not recognized for station address 01_02_03_04_FF_06:

- Sequence c)
  FF_FF_FF_FF_FF_FF_01_02_03_04_FF_FF_FF_FF_FF_FF_01_<complete sequence>

  11th byte (0xFF) is seen as the 11 byte of the partial pattern and is not recognized as the start of a complete sequence.

  Pattern search restarts looking for 6 bytes of 0xFF at 12th byte, but sees only 5.

Impact: The Ethernet controller will not exit Magic Packet mode if the Magic Packet sequence is placed immediately after other frame data which partially matches the Magic Packet Sequence.

Workaround: Place 1 byte of data that is not 0xFF and does not match any bytes of DA before the start of the Magic Packet sequence in the frame.

Because the Magic Packet sequence pattern search starts at the 3rd byte after DA, the Magic Packet Sequence can be placed at the start of the data payload as long as the second byte of the length/type field follows the above rule.

Fix plan: No plans to fix

eTSEC 12: MAC: Malformed Magic Packet Triggers Magic Packet Exit

Description: The Ethernet MAC should recognize Magic Packet sequences as follows:

Any Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of data payload. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFs, followed by 16 copies of the MAC's unique IEEE station address in the normal byte order for Ethernet addresses.

Once the Ethernet MAC has recognized a valid DA for one frame, it continues searching for valid 102-byte Magic Packet sequences through multiple frames without checking for valid DA on each frame. The only events that cause the MAC to go back to check for valid DA before checking for a Magic Packet sequence on new frames are:

1. A frame containing a recognized full Magic Packet sequence (with valid or invalid FCS)
2. Software disable of Magic Packet mode (MACCFG2[MPEN]=0)
3. MAC soft reset (MACCFG1[Soft_Reset]=1)

Impact: The Ethernet controller may exit Magic Packet mode if it receives a frame with DA not matching station address, or invalid unicast or broadcast address, but a valid Magic Packet sequence for the device.

Workaround: None

Fix plan: No plans to fix

eTSEC 13: No parser error for packets containing invalid IPv6 routing header packet

Description: A packet with an IPv6 routing header with the following invalid conditions will not be flagged as parser error.

- Segments Left field is greater than Header Extension Length field/2
- Header Extension Length field is not even
- Header Extension Length field is 0

As part of the pseudo-header calculation for the L4 checksum, the controller uses the last destination address from the routing header. It then calculates the L4 checksum and leaves the ETU bit in the RxFCB clear (marking this as a good checksum.) Since the above conditions constitute an invalid IVp6 routing packet, the checksum should not be marked as good and a parse error should be flagged instead.

The logic will do the following:

- Stop parsing the packet when the invalid IVP6 routing header is seen
- CTU bit is clear
- Indicate packet as bad to the filer via PID 1 bit 11. Note that no parser error will be indicated in the RxFCB (such as PER) or than that which is reported in the filer's PID 1 bit 11.

Impact: A packet with invalid IPv6 routing header will not be flagged as parse error. L4 checksum result (such as ETU) may be invalid.

Workaround: Use one of the following options:

- Set up the filter to detect the invalid IPV6 routing header using PID 1’s bit 11.
- If CTU bit is 0, consistency checks for IPv6 routing header packets must be performed in software, or skipped. If a packet does have a valid IPv6 routing header, then L4 checksum result, if enabled, can be considered as valid. Otherwise, software should consider the packet as malformed and should not use the packet's L4 checksum result stored in RxFCB.

Fix plan: No plans to fix

eTSEC 14: Receive pause frame with PTV = 0 does not resume transmission

Description: The Ethernet controller supports receive flow control using pause frames. If a pause frame is received, the controller sets a pause time counter to the control frame's pause time value, and stops transmitting frames as long as the counter is non-zero. The counter decrements once for every 512 bit-times. If a pause frame is received while the transmitter is still in pause state, the control frame's pause time value replaces the current value of the pause time counter, with the special case that if the pause control frame's pause time value is 0, the transmitter should exit pause state immediately. The controller does use the frame's pause time value to set the current pause time counter, but it then decrements the pause time counter before performing the compare to zero. As a result an XON (pause frame with PTV = 0), actually causes the transmitter to continue in pause state for 65,535 pause quanta, or 33,553,920 bit times.

Impact: A received pause frame with PTV = 0 causes the transmitter to pause for 65,535 pause_quanta. The expected behavior is for the controller to continue, or resume, transmission immediately. Note that the Ethernet controller always uses the value of the PTV register when generating pause frames. It never automatically generates a pause frame with pause time value of 0 when the receiver recovers from being above the RxFIFO threshold or below the free RxBDs threshold.

Workaround: To force an exit of pause state, use a pause frame with PTV value of 1 instead of 0.

Fix plan: No plans to fix

eTSEC 15: TxBD polling loop latency is 1024 bit-times instead of 512

Description: Register bit DMACTRL[WOP] defines the use of wait on poll when transmit ring scheduling algorithm is set to single polled ring mode. (TCTRL[TXSCHED]=00). When the use polling is selected by setting DMACTRL[WOP]=0, the poll to TxBD on ring 0 should occur every 512 bit-times. Due to the errata the poll occurs every 1024 bit-times.

Impact: The duration of the polling is twice as long as originally specified.

Workaround: None

Fix plan: No plans to fix

eTSEC 16: MAC: Rx frames of length MAXFRM or MAXFRM-1 are marked as truncated

Description: If MACCFG2[Huge Frame]=0 and the Ethernet controller receives frames which are larger than MAXFRM, the controller truncates the frames to length MAXFRM and marks RxBD[TR]=1 to indicate the error. The controller also erroneously marks RxBD[TR]=1 if the received frame length is MAXFRM or MAXFRM-1, even though those frames are not truncated.

No truncation or truncation error occurs if MACCFG2[Huge Frame]=1.

Impact: If MACCFG2[Huge Frame]=0, Rx frames of length MAXFRM or MAXFRM-1 are received normally, but RxBD[TR] is set to 1.

Workaround: Option 1:

Set MACCFG2[Huge Frame]=1, so no truncation occurs for invalid large frames. Software can determine if a frame is larger than MAXFRM by reading RxBD[LG] or RxBD[Data Length].

Option 2:

Set MAXFRM to 1538 (0x602) instead of the default 1536 (0x600), so normal-length frames are not marked as truncated. Software can examine RxBD[Data Length] to determine if the frame was larger than MAXFRM-2.

Fix plan: No plans to fix

eTSEC 17: Misfiled Packets Due to Incorrect Rx Filer Set Mask Rollback

Description: When the eTSEC Rx filer exits a cluster or AND chain that does not produce a match, it should roll back the mask to the value at the beginning of the chain or cluster. If that cluster or AND chain does not contain a Set Mask rule, however, the mask rolls back to the value prior to the previous Set Mask rule due to this erratum.

The following list provides an example of how a rule sequence should maintain a Mask for filer frame matching (the mask value is as it should be starting evaluation of the rule):

Rule #1: <Mask = default, 0xFFFF_FFFF>

Set Mask to 0x0000_8000 to search for frame data pattern that would match, as programmed by RQFPR's property field, and RQFCR[PID, Property ID].

Rule #2: <Mask = 0x0000_8000>

Look for a frame with status of having Broadcast address as the destination address.

Rule #3: <Mask = 0x0000_8000>

Start a Cluster of rules, grouped together for a special match.

Rule #4: <Mask = 0x0000_8000>

Set Mask to 0x0000_0210 inside Cluster, to search for frame data pattern that would match, as programmed by RQFPR's property field, and RQFCR[PID, Property ID].

Rule #5: <Mask = 0x0000_0210>

Inside Cluster, exit cluster and look for frame with status IPv4 header and UDP. Roll back mask to value at start of cluster.

Rule #6: <Mask = 0x0000_8000>

Set Mask (outside of cluster) to new value 0xFF00_FFFF.

Rule #7: <Mask = 0xFF00_FFFF>

Start of AND chain of 2 rules, the 1st rule, look for Destination Address 24-bits & Mask greater than RQPROP

Rule #8: <Mask = 0xFF00_FFFF>

End of AND chain, the 2nd rule, look for Destination Address low 24-bits & Mask greater than RQPROP. Roll back mask to value at start of chain.

Rule #9: <Mask = 0xFF00_FFFF>

Start of AND chain of 2 rules, the 1st rule, look for Source Address high 24-bits & Mask less than RQPROP

Rule #10: <Mask = 0xFF00_FFFF>

End of AND chain, the 2nd rule, look for Source Address low 24-bits & Mask less than RQPROP

Rule #11: <Mask = 0xFF00_FFFF>

etc.

The incorrect behavior in this example starts after rule #8:

Rule #8: <Mask = 0xFF00_FFFF>

End of AND chain, the 2nd rule, look for Destination Address low 24-bits & Mask greater than RQPROP. Roll back mask to value at start of chain.

After rule #8, the mask should roll back to the mask set by the last Set Mask rule outside the cluster (rule #6), but instead rolls back to the previous mask value (set by rule #1, and restored after cluster exit between rule 5 and 6).

Rule #9: <Mask = 0x0000_8000>

Start of AND chain of 2 rules, the 1st rule, look for Source Address high 24-bits & Mask less than RQPROP. Mask should be 0xFF00_FFFF.

Rule #10: <Mask = 0x0000_8000>

End of AND chain, the 2nd rule, look for Source Address low 24-bits & Mask less than RQPROP. Mask should be 0xFF00_FFFF.

Rule #11: <Mask = 0x0000_8000>

Mask should be 0xFF00_FFFF.

etc.

The AND chain of rule #9 and 10, or any rule that follows it, could falsely reject or misfile an incoming frame because the wrong mask is being applied.

Impact: The Filer can accept or reject a frame based on filer rule matching using incorrect mask.

Workaround: Use one of the following workarounds:

- Have all clusters or AND chains contain a Set Mask rule.
- After a stand alone Set Mask rule, the next cluster or AND chain in a sequence of filer rules should have immediately following it a repeat of the previous stand alone Set Mask rule.

Fix plan: No plans to fix

eTSEC 18: Incorrect frame data in L3+L4-only or L4-only parsing for FIFO mode

Description: eTSEC supports a variety of parsing options in FIFO mode (ECNTRL[FIFM]=1) based on the setting of RCTRL[PRSDEP] and RCTRL[PRSFM]. If PRSDEP=10, then PRSFM determines whether parsing is of L2+L3, or L3 only. If PRSDEP=11, then PRSFM determines whether parsing is of L2-L4, or L3-L4 only.

When PRSFM=0, the parser is supposed to skip all L2 parsing functions and assume the frame starts with an L3 header. If the 13th and 14th bytes of the L3 header, corresponding to the byte positions for the type field in an L2 header, happen to match the L2 type values for a control frame (0x8808), however, and RCTRL[CFA]=0, then the L2 parsing logic will corrupt some of the frame data sent to memory. Similarly, if the 13th and 14th bytes of the L3 header happen to match the VLAN L2 type (0x8100 or DFVLAN[Tag]), and RCTRL[VLEX]=1, then the L2 parsing logic will corrupt some of the frame data sent to memory.

Impact: When L3-only or L3+L4 header parsing is enabled in FIFO mode, the frame may be corrupted in memory if it appears to match the ethertype of a control or VLAN frame.

Workaround: Set RCTRL[CFA]=1 and RCTRL[VLEX]=0.

Setting CFA=1 prevents the eTSEC from corrupting a frame that appears to match a control frame ethertype (0x8808).

Setting VLEX=0 prevents the eTSEC from corrupting a frame that appears to match a VLAN ethertype (0x8100 of DFVLAN[Tag]).

Fix plan: No plans to fix

eTSEC 19: Excess delays when transmitting TOE=1 large frames

Description: The Ethernet controller supports generation of TCP or IP checksum in frames of all sizes. If TxBD[TOE]=1 and TCTRL[TUCSEN]=1 or TCTRL[IPCSEN]=1, the controller holds the frame in the TxFIFO while it fetches the data necessary to calculate the enabled checksum(s). Because the checksums are inserted near the beginning of the frame, transmission cannot start on a TOE=1 frame until the checksum calculation and insertion are complete.

For TOE=1 huge or jumbo frames, the data required to generate the checksum may exceed the 2500-byte threshold beyond which the controller constrains itself to one memory fetch every 256 eTSEC system clocks. This throttling threshold is supposed to trigger only when the controller has sufficient data to keep transmit active for the duration of the memory fetches. The state machine handling this threshold, however, fails to take large TOE frames into account. As a result, TOE=1 frames larger than 2500 bytes often see excess delays before start of transmission.

Impact: TOE=1 frames larger than 2500 bytes may see excess delays before start of transmission.

Workaround: Limit TOE=1 frames to less than 2500 bytes to avoid excess delays due to memory throttling.

When using packets larger than 2700 bytes, it is recommended to turn TOE off.

Fix plan: No plans to fix

eTSEC 20: Controller may not be able to transmit pause frame during pause state

Description: When the Ethernet controller pauses transmit of normal frames after receiving a pause control frame with PTV!=0, it should still be able to transmit pause control frames. The Ethernet controller, however, does not check whether the MAC is paused before initiating a start-of-frame request to the MAC. Once it has initiated a start-of-frame request, the Ethernet controller cannot initiate a pause control frame request until the normal frame completes transmission. Since the MAC will not transmit the normal frame until the pause time expires, this means the controller may be unable to transmit a pause frame while it is in pause state if there is a normal frame ready to transmit.

Impact: The Ethernet controller may be unable to transmit a pause frame during pause state if a normal frame is ready to transmit.

This applies to pause frame generation as a result of RxFIFO over threshold (ordinary flow control), free BDs below threshold (lossless flow control), or software-generated pause frame (TCTRL[TFC_PAUSE]).

Workaround: None

Fix plan: No plans to fix

eTSEC-A001: MAC: Pause time may be shorter than specified if transmit in progress

Description: When the Ethernet controller receives a pause frame with PTV!=0, and MACCFG1[Rx Flow]=1, it completes transmitting any current frame in progress, then should pause for PTV*512 bit times. The MAC, however, does not take the full transmission time of the current frame into account when calculating the Tx pause time, and may pause for 1-2 pause quanta (512-1024 bit times) less than the PTV value.

Impact: The eTSEC transmitter may pause transmission for up to 1024 bit times less than requested in a receive pause frame. If the PTV value does not contain at least 2 pause quanta worth of margin, this may lead to receive buffer overflows in the link partner.

Since the transmit pause does not take effect until after the current frame completes transmitting, the link partner's pause frame generator must already include the maximum frame size as margin when calculating the pause time value to use to prevent overflow of the receiver's buffers.

Workaround: Add 2 pause quanta to the pause time value used when generating pause frames to prevent receive buffer overflow.

Fix plan: No plans to fix

eTSEC-A002: Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame

Description: Ethernet standards define the minimum frame size as 64 bytes. The eTSEC controller also supports receiving short frames less than 64B, and can accept frames more than 16B and less than 64B if RCTRL[RSF] = 1. Frames shorter than 17 bytes are supposed to be silently dropped with no side-effects. There are, however, two scenarios in which receiving frames <= 2B cause erroneous behavior in the controller.

In the first scenario, if the last frame (such as an illegal runt packet or a packet with RX_ER asserted) received prior to asserting graceful receive stop (DMACTRL[GRS]=1) is <= 2 bytes, then the controller will fail to signal graceful receive stop complete (IEVENT[GRSC]) even though the GRS has successfully executed and the receive logic is completely idle. Any subsequent receive frame which is larger than 2 bytes will reset the state so the graceful stop can complete. An Rx reset will also reset the state.

In the second scenario, the parser and filer are enabled (RCTRL[PRSDEP] = 01,10,11). If a 1 or 1.5B frame is received, the controller will carry over some state from that frame to the next, causing the next frame to be parsed incorrectly. This in turn may cause incorrect parser results in RxFCB and incorrect filing (accept versus reject, or accept to wrong queue) for that following frame. The parser state recovers itself after receiving any frame >= 2B in length.

Impact: If software initiates a graceful receive stop after a 1- or 2-byte frame is received, the stop may not complete until another frame has been received.

A frame following a 1 or 1.5B frame may be parsed and filed incorrectly.

Workaround: For GRS scenario:

After asserting graceful receive stop (DMACTRL[GRS] = 1), initiate a timeout counter. The wait time is system and memory dependent, but a reasonable worst-case time is the receive time for a 9.6 Kbyte frame at 10/100/1000 Mbps. If IEVENT[GRSC] is still not set after the timeout, read the eTSEC register at offset 0xD1C. If bits 7-14 are the same as bits 23-30, the eTSEC Rx is assumed to be idle and the Rx can be safely reset. If the register fields are not equal, wait for another timeout period and check again.

MAX Rx reset procedure:

1) Clear MACCFG[RX_EN].

2) Wait three Rx clocks.

3) Set MACCFG2[RX_EN].

Fix plan: No plans to fix

IEEE1588_1: TxPAL timestamp uses TxBD snoop enable instead of Tx data

Description: The DMACTRL register contains two snoop enable bits for Tx: TBDSEN for buffer descriptors and TDSEN for frame data. Tx timestamp writes should use TDSEN to determine transaction snoop enable, but use TBDSEN instead.

Impact: If DMACTRL[TBDSEN] = 0, Tx timestamp writes to memory will not be snooped by the core regardless of the setting of DMACTRL[TDSEN].

If DMACTRL[TBDSEN] = 1, Tx timestamp writes to memory will be snooped by the core even if DMACTRL[TDSEN] = 0.

Workaround: Set DMACTRL[TBDSEN] = 1 if snooping of Tx timestamp writes to memory is desired.

Fix plan: No plans to fix

IEEE1588_2: Odd prescale values not supported

Description: The 1588 timer prescale register (TMR_PRSC) defines the timer prescale as follows: PRSC_OCK: Output clock division/prescale factor. Output clock is generated by dividing the timer input clock by this number. Programmed value in this field must be greater than 1. Any value less than 1 is treated as 2.

The output pulse (TSEC_TMR_PPn) width should be 1x the output clock width. For odd prescale values, the pulse width is 1.5x the output clock width instead.

Impact: Odd 1588 timer prescale values are not supported.

Workaround: Use only even timer prescale values.

Fix plan: No plans to fix

IEEE1588_3: Tx FIFO data parity error (DPE) may corrupt Tx timestamps if TMR_CTRL[TRTPE]=1

Description: If TMR_CTRL[TRPTE] = 1, the Ethernet controller writes Tx timestamp information for frames with TxFCB[PTP] = 1 to external memory in a PAL region. There is a queue of TxPAL addresses in the Ethernet controller that doesn't get cleared upon recovery from a Tx FIFO data parity error. As a result, if there were any frames with TxFCB[PTP]=1 in the Tx FIFO at the time the data parity error occurred (other than the frame with the error), a latent timestamp write for those frames may occur any time within the transmission of the first 8 TxBDs after DPE recovery. There may be up to three frames in the Tx FIFO in addition to the one with the parity error.

Example:

Suppose there are 4 frames in the Tx FIFO at time of the DPE, each with TxFCB[PTP] = 1 and each therefore using 2 BDs: BD0–BD7. When the DPE occurs, BD0 and BD1 are closed normally with an underflow indication, and BDs 2–7 are closed with an underrun indication, but TxPAL addresses for BD2, BD4 and BD6 are saved to a TxPAL queue and left valid. After the DPE recovery, the first 8 BDs transmitted trigger a write of invalid timestamp state to the TxPAL addresses of BD2, BD4 and BD6.

Impact: If TMR_CTRL[TRTPE]=1, data parity error may cause corruption of Tx timestamp data for PTP frames flushed due to the DPE.

Workaround: Transmit 8 or more non-TxPAL BDs which do not use any of the previously flushed PTP BDs, before allowing retransmission of those PTP BDs. These BDs must either be non-PTP frames (TxFCB[PTP]=0), or TMR_CTRL[TRTPE] must be 0. This can be done for example by:

- Keeping the ring(s) containing the flushed PTP BDs in halt while enabling other ring(s) containing at least 8 ready BDs. The 8 BDs can be for frames already programmed for transmission, or new ones for dummy frames in a queue enabled just for the purpose of clearing the TxPAL state.

OR

- Reordering the PTP BDs in the ring—moving them at least 8 entries down—so that least 8 other BDs in the ring are transmitted first.

In both cases, if the 8 BDs to be transmitted cannot be guaranteed to have TxFCB[PTP] = 0, then TMR_CTRL[TRTPE] must be set to 0 until those 8 BDs have been transmitted. Note that while TMR_CTRL[TRTPE] = 0, there will be no true Tx timestamp writes to memory for PTP frames, so it is preferable to ensure that only non-PTP frames are transmitted for the first 8 BDs.

Either workaround guarantees the true Tx timestamp write to memory for the flushed PTP frames (if enabled) will occur after the corrupted Tx timestamp write. Disabling TxPAL by setting TMR_CTRL[TRTPE] = 0 does not prevent the data corruption.

Fix plan: No plans to fix

IEEE1588_4: eTSEC 1588: Write to reserved 1588 register space causes system hang

Description: The IEEE 1588 control block contains a set of memory-mapped registers shared by all eTSEC controllers. These shared IEEE 1588 registers are located in the eTSEC1 controller memory space and support both read and write operations. The shared registers are located at the following addresses relative to the eTSEC1 base address 0x2_4000:

- 0xE20 (TMR_ADD)
- 0xE28 (TMR_PRSC)
- 0xE30 (TMR_OFF_H)
- 0xE34 (TMR_OFF_L)
- 0xE40 (TMR_ALARM1_H)
- 0xE44 (TMR_ALARM1_L)
- 0xE48 (TMR_ALARM2_H)
- 0xE4C (TMR_ALARM2_L)
- 0xE80 (TMR_FIPER1)
- 0xE84 (TMR_FIPER2)
- 0xE88 (TIMER_FIPER3)

Reads made to these addresses in the eTSEC3 memory space (0X2_6000) which are reserved, return 0's. Writes made to these addresses in the eTSEC3 memory space should be silently dropped, but instead cause a hang of the memory-mapped register IP interface; memory-mapped registers will not be accessible until the timeout mechanism implemented by the originating source of the transaction times out.

Impact: A programming error (write to reserved address space) may cause a hang.

Workaround: Do not write to 1588 shared register space within the eTSEC3 memory map.

Fix plan: No plans to fix

IEEE1588_5: 1588 alarm fires when programmed to less than current time

Description: The 1588 alarm mechanism operates purely on a less-than-or-equal-to comparison with the current time. If the alarm is programmed to a value less than the current time, it fires immediately, rather than waiting for the current time to wrap around and cross the alarm time.

This erratum only affects Alarm 2.

Impact: Alarm may fire before the intended time if armed when current time value is greater than alarm value.

Workaround: Ensure that the current time is less than the intended alarm time when programming TMR_ALARMn_H to arm the event.

Fix plan: No plans to fix

IEEE1588_6: IEEE 1588 accuracy can be adversely impacted in systems using multiple unsynchronized gigabit Ethernet ports

Description: In both RGMII and GMII Ethernet interface modes, all MACs use the GTX_CLK125 reference clock input to create the transmit clock that is used to transmit data from the MAC to the PHY. The MAC and PHY should be operated synchronously using a common clock reference although it is possible for the PHYs attached to these interfaces to transmit data across the Ethernet link at a slightly different frequency than they receive data from the MAC. This can occur if a PHY is configured as a slave PHY either manually or during the 1000Base-T Auto negotiation process. When configured as a slave, a PHY recovers the timing reference from the received link signal and uses this timing reference for transmit operations.

If the slave PHY's recovered timing reference has any frequency variations compared to the GTX_CLK125 clock that is used to transfer data from the MAC to the PHY, then the PHY transmitter will likely exhibit variations in delay due to the different clock frequencies. In systems that use any combination of two or more RGMII or GMII interfaces, it is possible for all of the PHYs attached to these individual interfaces to be configured as slave PHYs that have different reference frequencies. Although using the PHY's recovered output clock to drive the MAC's GTX_CLK125 reference clock enables one of the PHYs to operate synchronously with the MAC interface, it is not generally possible to synchronize all slave PHYs to their attached MACs since all MACs use a common GTX_CLK125 reference clock. This issue does not affect MACs configured in MII, RMII, or SGMII interface modes.

Impact: IEEE 1588 accuracy can be adversely impacted in systems using multiple unsynchronized gigabit Ethernet ports.

Workaround: Use one of the following workarounds to minimize delay variations within a PHY attached to the MAC interface:

- Use only one MAC configured in RGMII or GMII mode for passing IEEE 1588 messages and synchronize the MAC transmit interface to the PHY transmitter by using the PHY's recovered 125 MHz output clock as the GTX_CLK125 clock input. This allows the PHY to be configured as either a MASTER PHY or SLAVE PHY during the Auto negotiation process.
- Use one or more MAC interfaces in RGMII or GMII mode and force the attached MACs to be in MASTER PHY mode by writing to the appropriate PHY control registers. This will prevent the PHYs from Auto negotiating into SLAVE mode, and it allows a common 125 MHz timing reference to be supplied to all MACs and PHYs on the board.
- Use one or more MAC interfaces configured in MII, RMII, or SGMII modes for passing IEEE 1588 messages.

Fix plan: No plans to fix

IEEE1588-A001: Incorrect received timestamp or dropped packet when 1588 time-stamping is enabled

Description: When timestamping is enabled for all packets arriving on an Ethernet port (TMR_CTRL[TE] = 1 and RCTRL[TS] = 1), the port may fail to properly recognize the timestamp point for received frames. As a result, the timestamp value for the frame may be larger than the correct value, or the frame may be dropped.

Impact: For all modes except RMII, the timestamp value for a received frame may be larger than the correct value, or the frame may be dropped.

This erratum does not affect the operation of the TRIGGER_IN inputs, ALARM outputs, or FIPER outputs.

This erratum does not affect the operation of an Ethernet port when time-stamping is disabled (TMR_CTRL[TE]=0 and RCTRL[TS]=0); HRESET default is disabled.

Workaround: There is no workaround for the issue other than disabling receive time-stamping (RCTRL[TS]=0).

Fix plan: No plans to fix

GPIO 1: Invalid data read from GPDAT bits corresponding to pins configured as outputs

Description: GPIO pins can be configured as either inputs or outputs using the appropriate bits in the GPDIR register. Data read from GPDAT normally reflects the status of the corresponding GPIO pins. However, based on this erratum, reads to GPDAT will return invalid data for bits corresponding to pins configured as outputs. Note that although the correct value is driven on the pin, it is not verifiable by a read to GPDAT. For pins configured as inputs, reads to GPDAT will return valid data.

Impact: Reads to GPDAT return invalid data for bits corresponding to pins configured as outputs. Therefore, it is not possible to use software to read-modify-write the GPDAT register.

Workaround: Maintain a separate memory location which contains the value written to the GPIO outputs. When reading the status of the GPIO pins, read the status of the input pins from the GPDAT register, and read the status of the output pins from the separate memory location. Perform a bitwise OR between the two values to get the full status of the GPIO pins. Modify the value of the output pins as needed and write the result to both the GPDAT register and the separate memory location. For these read-modify-write sequences, another master must not modify GPDAT nor the separate memory location between the read and write operations.

Fix plan: Fixed in Rev. 1.1

I2C 1: Enabling I^{2}C could cause I^{2}C bus freeze when other I^{2}C devices communicate

Description: When the I^{2}C controller is enabled by software, if the signal SCL is high, the signal SDA is low, and the I^{2}C address matches the data pattern on the SDA bus right after enabling, an ACK is issued on the bus. The ACK is issued because the I^{2}C controller detects a START condition due to the nature of the SCL and SDA signals at the point of enablement. When this occurs, it may cause the I^{2}C bus to freeze. However, it happens very rarely due to the need for two conditions to occur at the same time.

Impact: Enabling the I^{2}C controller may cause the I^{2}C bus to freeze while other I^{2}C devices communicate on the bus.

Workaround: Use one of the following workarounds:

- Enable the I^{2}C controller before starting any I^{2}C communications on the bus. This is the preferred solution.
- If the I^{2}C controller is configured as a slave, implement the following steps:
  a. Software enables the device by setting I2CnCR[MEN] = 1 and starts a timer.
  b. Delay for 4 I^{2}C bus clocks.
  c. Check Bus Busy bit (I2CnSR[MBB])

```
if MBB == 0
                jump to Step f; (Good condition. Go to Normal
operation)
                else
                Disable Device (I2CnCR[MEN] = 0)
```

  d. Reconfigure all I^{2}C registers if necessary.
  e. Go back to Step a.
  f. Normal operation.

Fix plan: No plans to fix

PCIe 1: Reads to PCI Express CCSRs or local config space temporarily return all Fs

Description: When its link goes down the PCI Express controller clears all outstanding transactions and, if PEX_PME_MES_DISR[LDDD]=0 and PEX_PME_MES_IER[LDDIE]=1, sends a link down exception to the interrupt controller. Read transactions are cleared with an error indicator (transaction error to source for memory reads, return data all Fs for config reads). Write transactions are silently dropped.

If the controller is configured as an endpoint and receives a hot reset request, it also brings the link down and follows the same cleanup procedures as in an externally detected link down. Note that reads to PCI Express memory-mapped registers or config registers will return all Fs for the duration of the hot reset event, as well.

Impact: Reads to configuration, control and status registers in the memory-mapped or config space of PCI Express return all Fs during link down or hot reset event. During this time, Writes are handled normally, though the results of the write are not verifiable.

Workaround: If the configuration, control or status register cannot return all Fs as a legal value, but does, keep polling the register value until it is not all Fs. Note that PEX_PME_MES_DR, the detect register containing the link down detect bit, cannot return all Fs in normal operation.

If the configuration, control or status register can return all Fs as a legal value, and does, read another register which is known not to contain all Fs (e.g. PEX_IP_BLK_REV1) to confirm that the link is not in hot reset or link down cleanup, then reread the original register.

Once the link is confirmed down, register accesses can be enabled by writing 0x80C0_0000 to the PCI Express memory-mapped debug mode register at offset 0xF00 followed by polling the PEX_IP_BLK_REV1 register until it returns non-Fs to confirm that the reset of pending transactions is complete. Once software has completed any needed register accesses, it should return to normal function by writing 0x8040_0000 to the PCI Express debug register at offset 0xF00.

Be aware that while the PCI Express controller is in the debug mode that enables normal register accesses, new reads or writes to PCI Express are not canceled. Software must ensure that no new accesses are sent to PCI Express while this debug mode is enabled.

Fix plan: No plans to fix

PCIe 2: Excess correctable errors in receiving DLLPs and TLPs on x2 link

Description: The PCI Express specification allows for graceful recovery from bit errors on the interface through packet CRC error detection and recovery. The PCI Express controller, however, may flag a correctable error ("CED" in PCI Express device status register and "bad DLLP" or "bad TLP" in the PCI Express correctable error status register) based on an internal state error when receiving DLLPs and/or TLPs in x2 link width. When "bad TLP" correctable error status bit is flagged, the PCI Express controller NAKs the valid TLP received, which causes the remote PCI Express link partner to replay the TLP affected. In some cases, the frequency of these internal errors may be high enough to cause replay timeouts in the remote link partner. However, there is no link down being noticed.

Because DLLPs as well as TLPs can be affected, the errors may occur even if the controller is not receiving or transmitting TLPs (for example, during flow control initialization or updates).

This problem may occur only when the platform is running at frequencies of 500 MHz or higher. It only happens on the PCI Express controller 1 and 2.

Impact: PCI Express controller may observe excess correctable errors when trained as a x2 link. A PCI Express controller performance degradation of up to 10% may be observed.

Workaround: Use PCI Express controller 3 or lower platform frequency.

Fix plan: Fixed in Rev 1.2

PCIe-A001: PCI Express Hot Reset event may cause data corruption

Description: When the PCI Express controller is configured in EP Mode, if the controller detects an in-band Hot Reset event from its upstream device (either RC or Switch) before it finishes processing an inbound memory write TLP, the following may occur.

- TLP received right before the Hot Reset event may be discarded
- Data corruption may occur on the first inbound memory transaction received after the Hot Reset event.

Depending on the type of the first inbound memory transaction received after Hot Reset, data corruption may occur as below:

- If it is a memory write, the transaction may finish with data corruption at the target.
- If it is a memory read, the transaction may be decoded incorrectly and the return data might be incorrect.

Impact: This only affects devices with PCI Express controller configured in EP Mode.

An inbound memory write TLP received by the PCI Express controller in EP Mode may be discarded, if a Hot Reset event is detected while the controller is still in the middle of moving the payload data of this memory write TLP from its receiver buffer toward the packet destination. As a consequence, data corruption may also occur on the first inbound memory transaction received right after this “inbound memory write followed immediately by a Hot Reset event” sequence.

Note that after the data corruption occurs, the system will return to normal operating condition.

If the first inbound transaction received is a configuration cycle, after the above mentioned “inbound memory write followed immediately by Hot Reset event” sequence, the configuration cycle will finish normally, with no error. Since a Hot Reset event resets all the configuration space registers in any PCI Express EP controller, per PCI Express base specification requirements, the upstream RC must re-configure all these registers after the Hot Reset event. Therefore, with the normal PCI Express programming model, there are always configuration cycles before the upstream device can send memory transactions to downstream EP controllers, which means the data corruption scenario after the Hot Reset event might not happen for most applications. However, the Memory Write TLP received immediately before the Hot Reset event might still be discarded. End product designers must check against their application programming model and determine the actual impact and appropriate workaround adoption.

The above described error will not occur in any of the following conditions:

- If the last inbound memory transaction received immediately before the Hot Reset event is a memory read, or,
- If the system (PCI Express controller) is idle in its inbound path when the Hot Reset event occurs.

Workaround: When possible, before issuing the Hot Reset, the upstream RC or Switch should quiesce the system first to ensure no inbound traffic is flowing into the Freescale PCI Express EP controller. This prevents the above mentioned “inbound memory write followed immediately by Hot Reset event” sequence from occurring. The exact requirement and action required to quiesce the system is dependent on system, application and software used. For example, some requirements may include, but not be limited to, using a software semaphore at the RC system side to stop the new memory requests targeting the downstream Freescale PCI Express EP controller from the RC system’s software API layer or DMA controller.

Once such “quiescing system” actions have been finished, the RC system can send a configuration write cycle to clear the Memory Space bit in the Freescale PCI Express EP controller’s Command Register, followed by a several micro-second delay to allow all previously received inbound memory writes to propagate through the EP controller. A Hot Reset command can then be applied.

If an “inbound memory write followed immediately by Hot Reset event” sequence cannot be avoided, do the following. Right after the Hot Reset event, the upstream RC can issue a dummy memory write followed by another write or read to the read-only PEX_IP_BLK_REV1 memory-mapped register in the downstream Freescale device with PCI Express controller configured in EP Mode. The RC can then re-transfer the last memory write TLP that occurred right before the Hot Reset event and resume other normal traffic.

Fix plan: No plans to fix

PCI 1: Master-Abort issued incorrectly for outbound DAC with subtractive decode

Description: If an outbound command is issued by a PCI host and no response to the transaction occurs through the subtractive decoding cycle, then per the PCI specification a Master-Abort command should be issued by the host on the cycle subsequent to the subtractive decoding cycle. The device does not conform to the PCI standard for DAC transactions, and instead issues the Master-Abort one cycle earlier.

Impact: This device is not fully compliant with PCI rev2.2 specification regarding Master-Abort for DEVSEL not asserted, for DAC transactions. However, in most cases the subtractive decoding is used in error condition (when no device answers).

Workaround: Do not use subtractive decoding for DAC transactions other than error conditions.

Fix plan: No plans to fix

PCI 2: Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang

Description: As a master, the PCI IP block can combine a memory write to the last PCI double word (4 bytes) of a cacheline with a 4 byte memory write to the first PCI double word of the subsequent cacheline.

This only occurs if the second memory write arrives to the PCI IP block before the deassertion of $\overline{\text{FRAME}}$ for the first write transaction. If the writes are combined, the PCI IP block masters a single memory-write transaction on the PCI bus. If for this transaction, the PCI target asserts $\overline{\text{STOP}}$ during the last data beat of the transaction ($\overline{\text{FRAME}}$ is deasserted, but $\overline{\text{TRDY}}$ and $\overline{\text{IRDY}}$are asserted), the transaction completes correctly. A subsequent write transaction other than an 8-byte write transaction causes a hang on the bus. Two different hang conditions can occur:

- If the target disconnects with data on the first beat of this last write transaction, the PCI IP block deasserts $\overline{\text{IRDY}}$ on the same cycle as it deasserts $\overline{\text{FRAME}}$ (PCI protocol violation), and no more transactions will be mastered by the PCI IP block.
- If the target does not disconnect with data on the first beat of this last write transaction, $\overline{\text{IRDY}}$ will be deasserted after the first beat is transferred and will not be asserted anymore after that, causing a hang.

Impact: This affects 32-bit PCI target devices that blindly assert $\overline{\text{STOP}}$ on memory-write transactions, without detecting that the data beat being transferred is the last data beat of the transaction. It can cause a hang.

If the PCI transaction is a one data beat transaction and the target asserts $\overline{\text{STOP}}$ during the transfer of that beat, there is no impact.

Workaround: Hardware workaround:

Ensure that the PCI target device does not assert $\overline{\text{STOP}}$ during the last beat of a PCI memory write transaction that is greater than one data beat and crosses a cacheline boundary. It could assert $\overline{\text{STOP}}$ during the last data beat of the 32-byte cacheline or not assert $\overline{\text{STOP}}$ at all.

Software workarounds:

Set bit 10, the master disabling streaming (MDS) bit, of the PCI bus function register (address 0x44) to prevent the combining of discrete outbound PCI writes or the recombining of writes that cross the 32-byte cacheline boundary into a burst.

Set the PCI latency timer register (offset 0x0D) to zero. A value of zero is the reset value for this register, so keeping this register unmodified after reset prevents the PCI IP block from ever combining writes. It is not necessary to set the PCI latency timer register to zero if the MDS bit is set.

Fix plan: No plans to fix

SATA 1: Reads of one sector from hard disk may return less data than requested

Description: When the SATA controller issues a read DMA command and initiates a read of 512 bytes of data (one sector), a faulty hard disk may send less than the requested amount of data with a valid CRC and end the transmission of the data FIS, instead of sending the entire 512 bytes.

In this scenario, the SATA controller should perform a check for the total number of bytes it receives, as its DMA is working to write 512 bytes into memory but has received fewer bytes than that. However, it does not perform a check.

Instead, the SATA controller sets the command n completed (CCn) bit in the CCR (Command Completed Register) corresponding to the command n queue (CQn) bit in the CQR (Command Queue Register). This results in the HSTatus[CC] bit being set. No error bits are set in the HStatus register.

Impact: This would be of particular concern when more than one sector is being read from the device and the device sends fewer sectors than it should have sent.

Workaround: If the hard disk provides less data than requested by a command and then terminates that command with a good status FIS, then the command would be marked as complete.

In this, there are two situations:

1. If the hard disk indicates that the frame is transferred with error, the SATA controller does not have a mechanism to detect the length mismatch. In this scenario, the error detect mechanism will have to be implemented in customer’s software.
2. If the hard disk indicates that the frame is transferred with good status:
    a. If the data contain correct CRC, the SATA controller does not have a mechanism to determine the transfer length error. In this scenario, the error detect mechanism will have to be implemented in customer’s software.
    b. If the data contain incorrect CRC, then SError[T] should be set along with corresponding Command Error bit.

Fix plan: No plans to fix

SATA 2: SATA controller hangs when handling data integrity errors

Description: The SATA controller hangs when it faces data integrity errors due to CRC/Disparity/Decoding errors that cause changes in the host controller‘s internal state either due to DMA or through the PIO. When this happens, it must be brought offline and then online again to resume normal operation.

When a read/write DMA command is issued to the device and the driver/responder portions of the verification environment are configured to respond with a CRC/Decode/Disparity error, the command is issued in any one of the sixteen available command queue slots. The host detects the error and indicates the device error by setting bit 0 (DE0) of the DER (device error register) because only one device is connected to it. The host also sets the command error bit (CEn) for that particular command by setting the appropriate bit of the CER (command error register). However, the command complete bit setting always points to the slot zero command, even if command zero passed successfully.After this SATA controller hangs and does not issue any new commands.

Impact: The expected impact is expected to be minor as this is a rare scenario; under such a scenario, the SATA host will reset the link between the disk and the device, and software will just issue the commands again.This may impact performance if the software keeps track of the commands that were successfully completed and only issues bad commands again.

Workaround: None

Fix plan: No plans to fix

SATA 3: SATA BIST Activate FIS L bit is only valid in combination with the T bit

Description: This occurrence is a violation of the SATA Standard. The BIST Activate FIS ‘L’ bit should be valid on its own. Instead, it is only valid in combination with the ‘T’ bit.

Impact: The device cannot enter BIST-L mode with a standard FIS command.

Workaround: Use one of the following options:

- Issue a non-standard FIS command with both T and L bits set
- Use the internal PHY control register to set BIST-L mode

Fix plan: Fixed in Rev. 1.1

SATA 4: SATA Host does not acknowledge BIST Activate FIS and it immediately enters loopback mode.

Description: SATA Standard states that a BIST Activate FIS shall not be considered successfully transmitted until the receiver has acknowledged the reception of the FIS. The SATA Host controller does not acknowledge the BIST Activate FIS and immediately enters loopback mode.

Impact: Test equipment may hang while waiting for a response.

Workaround: Check system test equipment and ensure that it does not trigger on the FIS Receive Acknowledge. If it does trigger on this Acknowledge, the test equipment may hang.

Fix plan: No plans to fix

SATA 5: BIST-L mode is not enabled by BIST-L FIS

Description: The SATA controller cannot be put into BIST-L mode by BIST-L FIS.

Impact: The impact is not significant as alternative methods can be used to put SATA controller into BIST-L mode.

Workaround: None

Fix plan: Fixed in Rev 1.1

SATA 6: SATA: DMAT handling in SATA not as expected

Description: According to the SATA specification, during a DMA write to a device, the device may send a DMA Terminate (DMAT) primitive back to the host to abort the transmission. Upon receiving a DMAT primitive, the host should cease transfer by deactivating its DMA engine and completing the frame by sending a valid CRC and an EOF primitive. The host DMA engine will save its state at the point it was deactivated and at a later time, the device may choose to resume the transfer by transmitting another DMA Activate FIS or close it entirely.

However, when the device reissues the DMA Activate FIS to resume operation, the host does not resume normal operation.

Impact: On the reception of the DMA Activate FIS to resume the operation, the Host does not resume the DMA transfer from the point it stopped the last DMA operation. Instead, the Host is in a hang state as it does not have any context saved from previous transactions to resume the DMA transfer. It waits for either an intervention from the device or from software. If the device sends a Status FIS signaling Error, then the host will terminate the transaction. If the device does not send any additional FIS after the DMA Activate FIS, the host will wait until software interferes due to command timeout. The Host waits for the intervention of the software to service the interrupt and then reissues the pending commands after resetting the host-device link. This may have an impact on the performance of the Host Controller provided excessive DMAT primitives are issued by the device.

Workaround: As the software stores the context of all the issued commands, it can reset the link and resume the operation from the point of interrupt.

Fix plan: No plans to fix

SATA-A002: ATAPI commands may fail to complete

Description: When ATAPI drives, such as DVD or CD-ROM drives, are connected, commands that require setting the ATAPI 'A' bit (bit 5 of Word3) in the command header may fail to complete in the following scenario.

1. The drive is connected directly to SATA port and command slot other than 0 is used to issue ATAPI command
2. The drive is connected via PM (Port Multiplier) port and the command slot number is different from the PM port number on which ATAPI device is connected

This is because SATA controller misassigns the command number

Impact: Controller may not be able to detect or access ATAPI drives.

Workaround: For drive connected directly to SATA port:

- Use command slot 0 to issue ATAPI commands to ATAPI drives

For drives connected through PM Port:

- Use the command slot that matches the port number on which the ATAPI device is attached for command execution. For example, if an ATAPI device is attached to port 3 of the PM ports, use command slot 3 to send commands to this device.

Fix plan: No plans to fix

SEC 1: Non-compliant implementation of deterministic pseudo-random number generator

Description: There are two types of random number generators: true random number generators (True-RNG) and deterministic pseudo-random number generators (Pseudo-RNG).

- True-RNGs use an enviromental stimulus, such as the impact of temperature and voltage fluctuations on a ring oscillator, and combine it with feedback circuitry to produce a truly unpredictable random number.
- Deterministic Pseudo-RNGs use cryptographic algorithms, such as RSA or SHA, to produce a stream of random values from an initial seed value. Even if an attacker knows the method being used in a deterministic Pseudo-RNG, he cannot predict the next value because he does not know the seed value.

Although both types of RNGs can produce random numbers for a variety of cryptographic uses, the United States National Institute of Standards & Technology (NIST) only certifies deterministic Pseudo-RNGs. This is due to the inherent difficulty in proving that a True-RNG never outputs non-random data.

The RNG-B implements both a True-RNG and a deterministic Pseudo-RNG, based on SHA as described in FIPS 186-2, Appendix 3.1. This errata is due to the deterministic Pseudo-RNG implementation reversing the byte ordering associated with the XKEY operation described in FIPS 186-2, Appendix 3.1. Consequently, starting from a known seed, the RNG-B generates an equivalent-strength random output; however, this output is not what is defined for FIPS 182-2, Appendix 3.1 certification.

Impact: The impact of this errata is the system's inability to pass NIST RNG certification using the direct output of the RNG-B. Whenever NIST certification or use of NIST-compliant methods are required, the user must rely on a software implementation of a NIST approved deterministic Pseudo-RNG, with the associated software overhead. Generally NIST-compliant random numbers are only required for key generation/key exchange operations, which are relatively infrequent. The RNG-B can be used to seed the software deterministic Pseudo-RNG.

Workaround: As mentioned above, the RNG-B can be used to seed a software deterministic Pseudo-RNG when NIST-compliant random numbers are required. In the more common use cases, such as generation of random values to be used as Initialization Vectors (IVs) for security protocols such as IPsec, there is no requirement for NIST-compliant random numbers. In these cases, the RNG-B output can be used directly, with lower software overhead.

Fix plan: No plans to fix

SEC 2: Kasumi hardware ICV checking does not work

Description: SEC's execution units (EU) that generate integrity check values (ICVs) are also capable of comparing a received ICV with a calculated ICV. In the case of the KEU (Kasumi) F9 function, the SEC is not able to properly compare a received F9 MAC (another acronym for an ICV) with the calculated MAC.

Impact: The Kasumi algorithm produces a 128-bit integrity check value, which is shortened in the 3G protocol to a 64-bits message authentication code (MAC). To verify a received MAC, the user copies the received MAC to the KEU's IV data, which the SEC fetches (along with other parameters) in order to generate the calculated MAC. The KEU is supposed to compare the upper 64 bits of the calculated MAC with the 64 bits received MAC; however, it attempts to compare the full 128 bits and consequently always reports an ICV comparison failure, for example, integrity check comparison result (ICCR) returns binary "10."

Workaround: Users are not required to use the hardware ICV comparison feature. Rather than copying the 64 bits received MAC to the KEU's IV data, the user can merely have the SEC output the 128-bit MAC generated by the KEU, and software can perform the comparison of the upper 64 bits. Performing the MAC comparison in software takes only a few more CPU cycles than copying the received MAC to the IV data and reading the SEC's comparison result from the descriptor header.

Fix plan: No plans to fix

SEC-A001: Channel Hang with Zero Length Data

Description: Many algorithms have a minimum data size or block size on which they must operate. The SEC EUs detect when the input data size is not a legal value and signal this as an error. For most EUs, a zero byte length data input should be considered illegal, however the EUs do not properly notify the crypto-channel using the CHA of a data size error. Instead, the EUs wait forever for a valid data length, leading to an apparent channel hang condition.

Impact: When EUs detect illegal input data size, EUs wait forever for a valid data length, leading to an apparent channel hang condition.

Workaround: Option 1: Ensure that software does not create SEC descriptors to encrypt or decrypt zero length data.

Option 2: Use the SEC crypto-channel watchdog timer to detect hung channels. This is accomplished by enabling each channel's watchdog timer via the Channel Configuration Register CCRx[WGN]. This is a one time configuration. The timer stops when the channel completes a descriptor and restarts at zero each time the channel fetches a new descriptor. The default setting for the watchdog timer is 2^27 SEC clock cycles. If the timer expires, the channel will generate an interrupt and the Channel Status Register will show the cause as a Watchdog Timeout [WDT]. The channel hang is cleared by resetting the channel via CCR[RST]. The offending EU is reset automatically by the channel.

Fix plan: No plans to fix

USB 1: Read of PERIODICLISTBASE after successive writes may return a wrong value in host mode

Description: In the USB controller a new feature (hardware assist for device address setup) was introduced. This feature allows presetting of the device address in DEVICEADDR register before the device is enumerated, using a shadow register, to assist slow processors. The problem is that this mechanism, which is supposed to be functional only in device mode, is not blocked in host mode. DEVICEADDR register serves as PERIODICLISTBASE in host mode.

If PERIODICLISTBASE was set to some value, and later it is modified by software in such a way that bit 24 is set to 1, then wrong (previous) value is read back. However the USB controller will always read the correct value written in the register. ONLY the software initiated read-back operation will provide the wrong value.

Impact: No impact, if the software driver does not rely on the read-back value of the PERIODICLISTBASE register for its operation (practically there is no reason to do that).

Workaround: Write 0 twice to PERIODICLISTBASE before setting it to the desired value. This will clear the shadow register.

Fix plan: No plans to fix

USB 2: SE0_NAK issue

Description: When put into SE0_NAK test mode in device configuration, the USB controller may occasionally miss an IN token (not respond with a NAK token), if it was issued exactly on 125 microsec micro-frame boundary, when SOF is expected in functional mode.

Impact: This only affects the USB Test_SE0_NAK mode. No effect on normal operation.

Workaround: None

Fix plan: No plans to fix

USB 3: NackCnt field is not decremented when received NYET during FS/LS Bulk/ Interrupt mode

Description: The spec says that NakCnt should be decremented, whenever Host receives a NYET response to the Bulk CSPLIT and it should be reloaded when a start event is detected in the asynchronous list. This can happen in each micro-frame, or when the asynchronous schedule comes back from sleeping after an empty asynchronous schedule is detected.

The idea of the NAK counter is to keep trying until the counter reaches 0. When this happens, the controller just stops from issuing CSPLITS, until next micro-frame, where it reloads the counter and retries the CSPLIT again.

In the current implementation the controller does not decrement the NAK counter in this situation. If it receives a NYET, Host controller just advances to next Queue Head (QH) in the list and do not update the overlay area, later it will return and issue a new CSPLIT.

This could result in the host Controller thrashing memory, repeatedly fetching the queue head and executing transaction to the Hub, which will not complete until after the transaction on the classic bus completes.

The specification indicates that on receipt of a NYET handshake, the host controller should only adjust Cerr if it is the last CSPLIT transaction scheduled and halt the pipe when Cerr field transitions to zero. Since the Host controller hardware set the Cerr to the maximum value (3) every time it receives a NYET and stays in CSPLIT state so that the Cerr will never reach a zero value and hence the pipe will not be halted by the host controller.

Setting the Cerr to 3 every time it receives a NYET is a minor deviation from the specification. It will not cause a functional problem as a normal functioning hub. And it will eventually return something other than NYET and the transaction will complete.

The reclamation bit was being set to zero every time the host fetched the queue head with H bit set giving rise to empty list detection and the asynchronous list was not empty yet. This situation was leading the host to the asynchronous sleep state repeated times because the host was not paying any attention to NackCnt and RL.

Impact: The impact in the system is almost none. The Host will continuously do retries, instead of aborting when the NAK counter reaches zero. This may have a small penalty in the data bandwidth between Host and remaining attached Devices to the HUB.

Workaround: There is no direct software workaround, but the system software can cancel the transfer that is not reaching to the end after some time, and retry it later.

Fix plan: No plans to fix

USB 4: When an ACK or NAK is sent from the device in response to a PING, the CERR counter value is not being reset to the initial value

Description: When the Controller is acting as a Host, the host state machine needs to update the ping status in response to a PING. There are two situations which are successful packet handshaking: ACK and NAK. In these two cases, the Controller should reset the CERR field to its initial value. As the CERR field is not updated, there can be a situation where the qTD will be halted by this value reaching 0. Under this condition, the qTD token will be retired, even though at least one PING packet was successfully responded with ACK or NAK.

Impact: Some PING packets are retried, even though at least one PING packet was successfully responded with ACK or NAK.

Workaround: A software workaround for this issue is possible. If a value of 0 is used in field CERR of the dTD when building the data structures, the controller will retry the transaction continuously, and will not be retired due to consecutive errors. This change actually increases the chance for the transaction to complete.

Fix plan: No plans to fix

USB 5: In device mode, when receiving a Token OUT, if a Rx flush command is issued at the same time, to the same endpoint, the packet will be lost

Description: When receiving a Token OUT, the packet is lost if an Rx flush command is issued at the same time to the same endpoint. It reports ACK but the data is not copied to memory. Additionally, if the controller is in the stream disable mode (SDIS bit set a '1'), the endpoint is also blocked and NAKs any token sent to that endpoint forever. It is only applied to a USB device.

If the USB is configured as a peripheral, the controller normally does the flush when the software writes to the ENDPTFLUSH register of a Rx endpoint. The flush on the Rx buffer consists of DMA (drain side) reading all data remaining in the buffer until the Rx buffer is empty. The data is then thrown away.

If the Rx flush command is done while a packet is being received, the controller waits for the packet to finish and only then does the flush. As soon as the flush starts, the endpoint is unprimed, making the protocol engine (PE) NAK all incoming packets.

The protocol engine informs the endpoint control state machine that a packet has started by writing a TAG in the Rx buffer. As soon as the token is captured (knowing the endpoint and address), the protocol engine checks if the endpoint is primed or not. At this point, the protocol engine decides if the incoming packet will be ACKed or NAKed, even if the endpoint is unprimed during the packet reception.

The issue appears when the Rx flush command is set at the same time that the protocol engine writes the packet start TAG into the Rx buffer. At this moment, the endpoint control state machine does not have the packet because it has not read the packet start token yet. Thus, it starts the Rx flush sequence. At the same time by writing the packet start TAG into the Rx buffer, the endpoint is primed, so the protocol engine ACKs the packet at the end.

In this situation, the endpoint control state machine reads the Rx buffer at the same rate as the protocol engine writes the incoming data because all data is ignored. The state machine empties the Rx buffer and unprimes the endpoint. But the protocol engine ACKs the packet anyway because it only cares about prime when receiving an incoming token.

At the end of the packet, the protocol engine replies with ACK and writes the end of packet TAG into the Rx buffer. The endpoint is blocked because the protocol engine blocks it at the end of a non-setup transaction. The endpoint is unblocked by the endpoint control state machine as soon as all the data has been transfered into the system memory. As the endpoint control state machine never saw the start of packet, it never unblocks the endpoint. In stream disable mode, the only way to unblock the endpoint in this scenario is to switch to streaming mode.

Impact: Rx flush should not be used to clear an endpoint when acting as peripheral. When receiving a Token OUT, if an Rx flush command is issued at the same time to the same endpoint, the packet will be lost. For stream disable mode, the only way to unblock the endpoint in this scenario is switch to streaming mode.

Workaround: Do not use Rx flush feature when acting as peripheral.

Fix plan: No plans to fix

USB 6: In host mode, when the software forces a port resume by writing into the FPR bit of the portsc register, the port change detect interrupt bit is falsely fired

Description:	In host mode, a false "port change detect" interrupt is fired when the HCD (Host controller driver) resumes a suspended port by writing "1" to PORTSC[FPR] bit.
Impact:	An interrupt is falsely fired when the software forces a port resume. There is an extra overhead to deal with the mis-fired interrupt.
Workaround:	After setting PORTSC[FPR] and subsequent interrupt, the software should check the interrupt source, and clear USBSTS[PCI] bit, which corresponds to "port change detect" in Host mode.
Fix plan:	No plans to fix

USB 7: Transmit data loss based on bus latency

Description: When acting as a Device, after receiving a Token IN, the USB controller will reply with a data packet. If the bus memory access is not fast enough to backfill the TX fifo, it will cause an under-run. In this situation a CRC error will be introduced in the packet and the Host will ignore it. However, when an underrun happens, the TX fifo will get a flush command. This situation may cause an inconsistence in the TX fifo controls, leading to a possible data loss (a complete packet or sections of a packet can be never transmitted). This situation may also happen if the software issues a TX flush command.

Impact: When the USB controller is configured as a device, it can not be used in the stream mode due to this erratum. Therefore, the USB external bus utilization is decreased.

Workaround: A valid software workaround is to disable the stream mode by setting USBMODE[SDIS] bit. This can avoid the issue at the expense of decreased USB external bus utilization.

Fix plan: No plans to fix

USB 8: Priming ISO over SOF will cause transmitting bad packet with correct CRC

Description: In device mode, a priming operation performed by software while an IN token is being received (usually just after the SOF) can lead to an invalid transfer in the next frame or an abortion of a transfer that should not be canceled. This can only happen for Isochronous IN transfers.

This scenario can happen when the Device controller receives an IN token from the host and the protocol engine has not been primed yet, but the software has already performed the priming operation. This can occur due to latency between the setting of the prime bit for the specific endpoint and the exact moment when the protocol engine recognizes that it has been primed. The root cause is that in the current implementation, the priming operation inside the protocol engine only occurs at the time of SOF reception.

Impact: Two problems can arise as a result of this issue:

- The data that is being written into the Tx RAM is read while the device sends a zero length packet (since it is not primed). Therefore, the data cannot be transferred to the next frame, and a short packet is sent the next time the host asks for data.
- After sending the zero length packet, the protocol engine informs the DMA state machine that the transfer has been completed, so the respective dTD is updated. This also causes an error because the total number of bytes transferred is not equal to zero. Therefore, this transfer is also lost.

Workaround: There is no workaround.

Fix plan: No plans to fix

USB 9: Missing SOFs and false babble error due to Rx FIFO overflow

Description: When in Host mode, if an Rx FIFO overflow happens close to the next Start-of-Frame (SOF) token and the system bus (CCB) is not available, a false frame babble is reported to software and the port is halted by hardware. If one SOF is missed, the Host controller will issue false babble detection and SOFs will no longer be sent. If more than 3.125 ms are elapsed without SOFs, the peripheral will recognize the idle bus as a USB reset.

Impact: If this scenario occurs, it will degrade performance and have system implications. The Host will have to reset the bus and re-enumerate the connected device(s).

Workaround: Reset the port, do not disable the port, on which the babble is detected.

Fix plan: No plans to fix

USB 10: No error interrupt and no status will be generated due to ISO mult3 fulfillment error

Description: When using ISO IN endpoints with MULT = 3 and low bandwidth system bus access, the controller may enter into a wait loop situation without warning the software. Due to the low bandwidth, the last packet from a mult3 sequence may not be fetched in time before the last token IN is received for that microframe/endpoint.

Impact: This will cause the controller to reply with a zero length packet (ZLP), thus breaking the prime sequence. The DMA state machine will not be warned of this situation and the controller will send a ZLP to all the following IN tokens for that endpoint. The transaction will not be completed because the DMA state machine will be waiting for the unprimed TX complete command to come from the Protocol Engine.

Workaround: If this scenario occurs, use MULT = 2.

Fix plan: No plans to fix

USB 11: Wrong value read in BURSTSIZE[TXPBURST] and BURSTSIZE[RXPBURST] registers

Description: When reading the BURSTSIZE[TXPBURST] and BURSTSIZE[RXPBURST] registers , the read value is always 5'b10000, regardless of the previously programmed value in either of these registers.

Impact: The read value does not match the previously written value in each of these registers.

Workaround: None. The actual value written in the register is correct and the issue here is a register read problem.

Fix plan: No plans to fix

USB 12: CRC not inverted when host under-runs on OUT transactions

Description: In systems with high latency, the HOST can under-run on OUT transactions. In this situation, it is expected that the CRC of the truncated data packet to be the inverted (complemented), signaling an under-run situation.

Impact: Due to this erratum, the controller will not send this inverted CRC. Instead, it sends only one byte of the inverted CRC and the last byte of payload.

It is unlikely but remotely possible that this sent CRC to be correct for the truncated data packet and the device accepts the truncated packet from the host.

Workaround: Setting bigger threshold on TXFILLTUNING[TXFIFOTHRES] register might solve the under-run possibility and thus avoiding truncated packets without the expected inverted CRC.

However, this would not solve the inverted CRC issue by itself.

Fix plan: No plans to fix

USB 13: Core device fails when it receives two OUT transactions in a short time

Description: In the case where the Controller is configured as a device and the Host sends two consecutive ISO OUT (example sequence: OUT - DATA0 - OUT - DATA1) transactions with a short inter-packet delay between DATA0 and the second OUT (less than 200 ns), the device will see the DATA1 packet as a short-packet even if it is correctly formed. This will terminate the transfer from the device's point -of-view, generating an IOC interrupt. However, DATA0 is correctly received.

Impact: If this scenario occurs, the clear command from the protocol engine state machine to the protocol engine data path is not sent (and internal byte count in the data path module is not cleared). This causes a short packet to be reported to the DMA engine, which finishes the transfer and the current dTD is retired.

Workaround: None

Fix plan: No plans to fix

USB 14: USB Controller locks after Test mode "Test_K" is completed

Description: When using the ULPI interface, after finishing test mode "Test_K," the controller hangs. A reset needs to be applied.

Impact: No impact if reset is issued after "Test K" procedure (it should be issued according to the standard).

Workaround: None

Fix plan: No plans to fix

USB 15: NAK counter decremented after receiving a NYET from device

Description: When in host mode, after receiving a NYET to an OUT Token, the NAK counter is decremented when it should not.

Impact: The NAK counter may be lower than expected.

Workaround: None

Fix plan: No plans to fix

USB-A001: Multiple dTDs can cause USB device controller unprimed an end point

Description: After executing a dTD, the device controller executes a final read of the dTD terminate bit. This is done in order to verify if another dTD has been added to the linked list by software right at the last moment.

It was found that the last read of the current dTD is being performed after the interrupt was issued. This causes a potential race condition between this final dTD read and the interrupt handling routine servicing the interrupt on complete which may result in the software freeing the data structure memory location, prior to the last dTD read being completed. This issue is only applied to a USB device controller.

This erratum occurs very rarely because if the T bit of the next link pointer of the dTD that is being retired was set to'1' when software set it up then one of the following cases will be true:

1. If software has not updated the next link pointer of the only/final dTD since it linked it in to the link list then the device has the correct next link pointer information and no issue under this condition.
2. If software updated the next link pointer of the final dTD in a link list before the device performs the queue head over lay for the final dTD, then the device controller will have the correct next link pointer information and controller works as it supposed to be.
3. If software updated the next link pointer of the dTD after the device performs the queue head overlay and the interrupt handling routine has a higher latency than the controller read latency of the dTD then the device will read the correct information. For example if the worst case delay for the device to access the dTD is 1 micro second and the minimum interrupt handling routine latency is 5 micro seconds then the device will always complete the final read before the dTD is changed and so no effect to the normal operation.
4. If software updated the next link pointer of the dTD after the device performs the queue head overlay and if the interrupt handling routine has a lower latency in comparison to the controller read access of the dTD then software will update the retired dTD before the device does its final read, then the device will not have the correct information.

Impact: Due to the nature of the logic bug, it only occurs for the multiple dTDs with the IOC bit set. In this case, if the latency handling the interrupt is too long, the following might occur:

1. The USB controller could assert the interrupt when it completes dTD1.
2. Proceed to execute the transfer for dTD2.
3. By the time the controller has just updated dTD2 Active field to inactive, the interrupt handling routine finishes processing the interrupt for dTD1, checks dTD2 and finds that it has completed and so re-allocates both data structures.
4. The controller then re-reads dTD2 and finds corrupted data. If the T bit is zero or the Active field in non active then the controller will not re-prime.

As far as USB device controller is concerned, the time between sending the interrupt and initiating the final read of the dTD is 2 platform clocks. Freescale has not duplicated this issue on any silicon and no customers have reported this issue.

Workaround: The workaround is a combination hardware and software solution. The first 2 steps are needed for all the devices. Only one of the software workarounds in the Step 3 is needed.

1. Only configure one USB controller as the USB device controller.
2. Write Global Utilities Memory offset 0xE0F60 to 0
3. Use one of these two software workarounds.
    - The USB device controller driver should be written in the following way:

1. Create a pool of memory for dTDs from a system memory at the beginning of the USB initialization process. Pre-allocate a dTD buffer ring from this pool of the memory.
2. Only destroy the memory after the system is disconnected from the system.
3. Allocate a new dTD memory in a sequential manner from the pool of free memory space from substep 1. when it is needed.
4. Return the dTD memory to the pool of the memory space from substep 1. when the transaction is completed. Do not clear the memory.
5. In this way, the new released dTD memory will not be used/overwritten in the next dTD. Therefore, there should be enough time for the controller to read the memory before software overwriting the memory.

- For a bulk, interrupt, or control end point, the software can check the endpoint status register to see whether the endpoint is primed when the end point keeps sending NAK for a while. The software can update the endpoint queue head to point to the right dTD and set the appropriate endpoint prime bit in the ENDPTPRIME register. The device controller will prime in response. This solution does not work for an ISO endpoint.

Fix plan: No plans to fix

USB-A002: Device does not respond to INs after receiving corrupted handshake from previous IN transaction

Description: When configured as a device, a USB controller does not respond to subsequent IN tokens from the host after receiving a corrupted ACK to an IN transaction. This issue only occurs under the following two conditions:

1. An IN transaction after the corrupted ACK
2. The time gap between two IN tokens are smaller than the bus time out (BTO) timer of the USB device controller

Under this case, for every IN token that arrives, the bus timeout counter is reset and never reaches 0 and a BTO is never signaled. Otherwise, the USB device times out and the device controller goes to an idle state where it can start to respond normally to subsequent tokens. .

Impact: There are two cases to consider:

1. CERR of the Queue Element Transfer Descriptor (qTD) is initialized to a non-zero value by the host driver

   This is what happens in the majority of use cases. The maximum value for CERR is 3. A host driver is signaled/interrupted after CERR is decremented to 0 due to the transaction errors. In this case, the host driver has to process the transaction errors. Most likely, the host driver will reset this device. Furthermore, the BTO timer of the USB device could time out due to time needed for the USB host to process the transaction errors.

2. CERR of the qTD is initialized to zero by the host driver

   In this case, the host driver does not process any transaction error. However, based on USB 2.0/EHCI specification, the host controller is required to maintain the frame integrity, which means that the last transaction has to be completed by the EOF1 (End Of Frame) point within a (micro) frame. Normally, there should be enough time for a USB device to time out.

Workaround:

1. CERR of the qTD is initialized to a non-zero value

   No workaround is needed since the USB host driver will halt the pipe and process the transaction errors

2. CERR of the qTD is initialized to zero and if
   a. The USB controller is configured as a full/low-speed device.

      No workaround is needed since USB 2.0 specification requires a longer idle time before a SOF (Start of Frame) than the BTO timer value. The USB device times out at the end of the next (micro) frame at most.

   b. The USB controller is configured as a high-speed device.

      No workaround is needed if the data length of the next transaction after the corrupted ACK is 32 bytes or longer. The USB device times out at the end of the next (micro) frame at most.

   c. The USB controller is configured as a high-speed device.

      No workaround is needed as long as the Host_delay in the USB host system is 0.6 us or longer. The USB device times out at the end of the next (micro) frame at most.

   d. The USB controller is configured as a high-speed device.

A workaround is needed if the data length of the next transaction after the corrupted ACK is less than 32 bytes and the Host_delay in the USB host system is less than 0.6 us. Under this condition, a transfer in a device controller does not progress and the total bytes field in the dTD (device transfer descriptor) remains static. The software on a device controller could implement a timer routine for an IN endpoint to monitor whether a transfer is progressing. If it is not, the timer routine can cancel the transfer and reset the device by setting the USBCMD[RST] bit. However, this is a rare case. No such case has been reported.

Fix plan: No plans to fix

USB-A003: Illegal NOPID TX CMD issued by USB controller with ULPI interface

Description: During the USB reset process (speed negotiation and chirp), if the protocol engine sends Start of Frame (SOF) commands to the port control, the port control filters out those SOFs. However, at the end of reset (end of chirp back from Host), when the protocol engine sends a SOF, the ULPI port control sends the SOF to the PHY before sending the update OpMode command. This results in an invalid packet being sent on the line. The invalid packet in ULPI protocol is a NOPID transmit command immediately followed by a STP pulse. This failure happens less than 0.1% of time.

Impact: Due to this erratum, some ULPI PHY's lock up and do not accept any additional data from USB host controller. As PHY is locked up, there cannot be any communication on the USB Interface.

Workaround: Do not enable USBCMD[RS] for 300 uSec after the USB reset has been completed (after PORTSCx[PR] reset to 0). This ensures that the host does not send the SOF until the ULPI post reset processing has been completed.

Fix plan: No plans to fix

USB-A005: ULPI Viewport not Working for Read or Write Commands With Extended Address

Description: It is not possible to read or write the ULPI PHY extended register set (address >0x3F) using the ULPI viewport.

The write operation writes the address itself as data, and a read operation returns corrupted data. A Controller lock up is not expected, but there is no feedback on this failed register access.

Impact: Read or Write Commands With Extended Address does not work through ULPI Viewport register.

Workaround: None

Fix plan: No plans to fix

USB-A007: Host controller fails to enter the PING state on timeout during High Speed Bulk OUT/DATA transaction

Description: For High-speed bulk and control endpoints, a host controller queries the high-speed device endpoint with a special PING token to determine whether the device has sufficient space for the next OUT transaction. The mechanism avoids using bus time to send data until the host controller knows that the endpoint has space for the data.

If a timeout occurs after the data phase of an OUT transaction, the host controller should return to using a special PING token. However, due to this erratum, the host controller fails to enter the PING state and instead retries the OUT token again.

Impact: The PING flow control for the high-speed devices does not work under this condition. Therefore, some USB bandwidth could be wasted. However, a timeout response to Out/Data or PING transactions is an unexpected event and should only occur if the device has detected an error and so should be rare.

Workaround: None

Fix plan: No plans to fix

ESDHC 1: Read/Write Multiple command when block count is 1

Description: When the driver sends multiple read/write commands to the card and the block count is 1, with auto CMD12 enabled, eSDHC does not send auto CMD12 at the end of the transfer.

Impact: This erratum adds some overhead for the host controller software. In this state, the driver should send CMD12 and only then can it send another read/write command to the device.

Workaround: For a single block write, the driver should send the WRITE_BLOCK command (CMD24). For a single block read, the driver should send the READ_SINGLE_BLOCK command (CMD17).

Fix plan: No plans to fix

ESDHC 2: Force Event Register

Description: If any of the events in the interrupt status register are cleared in the previous writing of the interrupt status register, the event cannot be forced in the next (first) access to the force event register. This event will be forced only in the second access to the force event register.

Impact: There is an overhead for the host driver to write twice to the force event register.

Workaround: If the driver uses this feature, the IPGEN bit in the system control register should be high or the host driver should write twice to the force register to force the event.

Fix plan: No plans to fix

ESDHC 3: eSDHC reads more data than expected from the system

Description: When using the internal DMA to write data to the card, it is possible for the eSDHC to read more data than expected from the system address. This depends on the value of the WR_WML field in the watermark level register and on the DVS and SDCLKFS fields in the system control register.

Impact: After the DMA has stopped, the value of the DMA system address register should be the next system address of the next contiguous data position. Due to this issue, sometimes the value of this register is the next system address of the next contiguous data position plus the write burst length (WR_WML x 4).

This behavior can be problematic when the next contiguous address is not an accessible address. In this case, the eSDHC may have a transfer error and indicate a DMA error event.

Workaround: If the eSDHC DMA is used, ensure that the next contiguous address is an accessible address.

Fix plan: No plans to fix

ESDHC 4: eSDHC indicates AUTO CMD12 interrupt later than expected

Description: When the IPGEN bit in the system control register is cleared, and an error occurs during the response to AUTO CMD12, the eSDHC does not indicate an AUTO CMD12 interrupt until the IPGEN bit is set to 1 or until the host driver sends another command to the card. At this time, the AUTO CMD12 error status register gets the correct value, but the eSDHC does not indicate an AUTO CMD12 interrupt in the interrupt status register.

Impact: After sending multiple read/write commands, when AUTO CMD12 is enabled, the host driver does not know if an error occurred in response to the stop transmission command (CMD12).

Workaround: Do not clear the IPGEN bit in data transfer with the AUTO CMD12 bit set.

Fix plan: No plans to fix

ESDHC 5: Data corruption during write with pause operation

Description: In the write with pause operation, after data transfer is resumed, the eSDHC corrupts one word (32 bits) in the middle of the transfer and writes the word to the card. This behavior occurs when the HCKEN bit in the system control register is cleared.

Impact: When the above situation occurs, a wrong value is written to the card.

Workaround: Do not clear HCKEN bit during a DMA transfer. Always set HCKEN bit to 1.

Fix plan: No plans to fix

ESDHC 6: Cannot initiate non-data command while data transfer is in progress

Description: According to the SD physical specification, the host driver can issue CMD0, CMD12, and CMD13 when the data lines are busy during data transfer. The host driver should then send these commands during a block gap and afterwards the host driver is able to resume the transfer. However, the related command bits are not protected during the block gap and thus data transfer cannot be resumed. The transfer context would be corrupted if data transfer is resumed.

Impact: The host driver cannot initiate non-data transfer commands while a data transfer is in progress.

Workaround: None

Fix plan: No plans to fix

ESDHC 7: Incorrect data is written to a card after transfer paused

Description: For a write operation, if the host controller pauses the transfer, and there is no more data in the internal FIFO when transfer stops, an intermediate channel to write data to the card, loads dirty data and corrupts the beginning bytes of the next block when data transfer is resumed.

Due to this defect, the software may not reliably pause the write transfer if the write from system side is slow. That is, when the request to stop at block is sent out, if the system side is not writing data for the next block comparing the card side, data corruption occurs.

Impact: This issue only occurs for a write pause, and if suspend command is sent out to the card, this issue also disappears. In real applications, the requirement to pause and resume a write transfer is rare, so such an issue is not critical.

Workaround: The software workaround is to check the system side to confirm the system side has already written all the data blocks that are to write to the card prior to write pause, and then it is allowed to stop the transfer.

Fix plan: No plans to fix

ESDHC 8: CRC might be corrupted for write data transfer if it is preceded by read transfer

Description: Software writing to the Transfer Type configuration register (system clock domain) can cause a setup/hold violation in the CRC flops (card clock domain), which can cause write accesses to be sent with corrupt CRC values. This issue occurs only for write preceded by read.

Impact: A write data transfer that follows read may have corrupted CRC content. Software needs to reset the data path or shut off the card clock for such a scenario.

Workaround: Use one of the following options:

- Set SYSCTL[RSTD] (software reset) bit after each read operation is done (transfer complete) and reconfigure all related registers.
- After a read operation, use the following steps:
  a. Clear bits 2–0 of the system control register (PEREN, HCKEN, IPGEN).
  b. Wait for bit 7 or 6 of the present state register to be set, which means either the card clock or the internal baud rate clock stops.
  c. Send either CMD13 to poll status or CMD24 or CMD25 to launch write operation.

Fix plan: Fixed in Rev. 1.1

ESDHC 9: Data End Bit Error (DEBE, bit 22 in the interrupt status register) incorrectly set if Card Interrupt is driven in SDIO 4-bit mode

Description: Data End Bit Error (DEBE, bit 22, in the interrupt status register) is incorrectly set if Card Interrupt is driven in SDIO 4 bit mode. eSDHC monitors the end bit at CRC status. For 4-bit mode and 8-bit mode (MMC card only), if any of the data lines is “0” at the end bit cycle, the end bit error occurs. This assumption is correct for MMC/SD cards, but it does not work for SDIO cards. For a single block write or a predefined number of write blocks, the interrupt may occur during the CRC status stage. The “0” on DAT1 (which may occur due to SDIO interrupt) will then lead to a false end bit error.

Impact: False DEBE interrupt may be issued to the core, so core interrupted due to false error which is a software overhead.

Workaround: For SDIO card block write, if both card interrupt status and end bit error status are set, the software needs to ignore the end bit error.

Fix plan: No plans to fix

ESDHC 10: During multi-write operation, unexpected Transfer Complete (IRQSTAT[TC] register) bit can be set

Description: Invalid Transfer Complete (IRQSTAT[TC]) bit could be set during multi-write operation even when the BLK_CNT in BLKATTR register has not reached zero. Therefore, Transfer Complete might be reported twice due to this erratum since a valid Transfer Complete occurs when BLK_CNT reaches zero.

Impact: The IRQSTAT[TC] status bit is not reliable during the multiple block transfer. The software driver needs to send additional command to check the card status to ensure transfer complete, in case false TC is reported.

Workaround: Ignore the IRQSTAT[TC] register bit if BLKATTR[BLK_CNT] register field is confirmed to be non-zero.

If block count is zero, it is more robust to poll card status using CMD13 for MMC/SD cards or CCSR[Ready Flags] for SDIO.

Fix plan: No plans to fix

ESDHC 11: eSDHC interrupt not negated in case of card insertion/removal

Description: The interrupt to processor that generates from eSDHC due to card insertion or card removal does not negate even when the corresponding bit in IRQSTAT status register is cleared by the processor. The CINS or CRM bit in the IRQSTAT register is cleared, but eSDHC keeps interrupt signal asserted.

Impact: The impact is low because the card does not need to be inserted or removed very often.

Workaround: Upon receiving the card insertion interrupt, the software should disable the CINS interrupt in IRQSIGEN and enable the CRM interrupt in IRQSIGEN. Conversely, upon receiving the removal interrupt, the software should disable the CRM interrupt in IRQSIGEN and enable the CINS interrupt in IRQSIGEN.

Fix plan: No plans to fix

ESDHC 12: Unable to issue CMD12 if SD clock shuts off due to slow system access

Description: If the eSDHC issues a data transfer, but system side fails to access its internal buffer as quickly as the card, and the buffer fills up for a read transfer or empties for a write transfer, the eSDHC will stop the card clock to avoid buffer overrun (for read) or underrun (for write). If CMD12 (with XFERTYP[CMDTYP] = 3, abort) is issued to stop the transfer, the CMD12 never makes it onto the SD/MMC interface due to this erratum.

Impact: There is software overhead to initiate dummy write/read accesses to buffer for switching on the SD clock to issue CMD12.

Workaround: A software workaround may be implemented to check the card clock status on issuing CMD12 to abort the data transfer. If the clock is stopped, several accesses need to be applied to the buffer to activate the clock.

Upon sending CMD12 to abort the data transfer, poll the card clock status to see if it is stopped. If so, write XFER_TYPE register to issue the CMD12 and make several accesses (write or read depending on the current transfer direction) to the buffer to change the buffer state. Doing this will restore the clock, then send the command.

Fix plan: No plans to fix

ESDHC 13: Manual Asynchronous CMD12 abort operation causes protocol violations

Description: There may be protocol violations if a manual (software) asynchronous CMD12 is used to abort data transfer. Due to this erratum, the eSDHC controller continues driving data after a manual (software) asynchronous CMD12 is issued. Therefore, it may cause a conflict on the data lines on the SD bus.

Impact: Manual asynchronous CMD12 to terminate data transfer cannot be used.

Workaround: Do not issue a manual asynchronous CMD12. Instead, use a (software) synchronous CMD12 or AUTOCMD12 to abort data transfer.

For a manual synchronous CMD12, the following steps are required:

1. Set PROCTL[SABGREQ] = 1
2. Wait for IRQSTAT[TC] bit set, or Transfer Complete Interrupt
3. Set IRQSTAT[TC] = 1
4. Issue CMD12 after checking PRSSTAT[CIHB] = 0
5. Set both SYSCTL[RSTD] and SYSCTL[RSTC]

Fix plan: No plans to fix

ESDHC 14: PRSSTAT[CIDHB] is not reliable for commands with busy (R1b)

Description: PRSSTAT[DLA] (Data Line Active) is not reliable for commands, (such as CMD6, CMD7, CMD12, CMD28, CMD29, or CMD38), with busy signal. DLA affects PRSSTAT[CIDHB] (Command with Data Inhibit). Therefore, a software driver may not know the busy status in DLA/CIDHB and if it issues next command with data or R1b, there might be contention on SD bus and this might create data CRC error.

Impact: The PRSSTAT[CIDHB] and PRSSTAT[DLA] bit may not be used for commands with busy.

Workaround: Driver should read the card status, using SEND_STATUS command(CMD13), to check busy de-assertion before issuing next command which uses data line.

Fix plan: No plans to fix

ESDHC 15: The $\overline{\text{SDHC\_WP}}$ signal polarity is reversed

Description: GENCFGR[SDHC_WP_INV] does not affect the $\overline{\text{SDHC\_WP}}$ polarity due to this erratum. The system behaves as if GENCFGR[SDHC_WP_INV]=1 in this case. Regardless of GENCFGR[SDHC_WP_INV], the card is write protected if $\overline{\text{SDHC\_WP}}$ is 0 (and then PRSSTAT[WPSPL] is 1).

Impact: The write protected pin polarity does not follow the SD card standard for Rev 1.0 silicon.

Workaround: Put an extra inverter for the $\overline{\text{SDHC\_WP}}$ signal on the board if an active-high $\overline{\text{SDHC\_WP}}$ signal is desired.

Fix plan: Fixed in Rev 1.1

ESDHC 16: eSDHC cannot finish write operation after continuing from Block Gap Stop

Description: After stop at block gap in write operation, when the transfer is continued (by setting PROCTL[CREQ]), the data transfer cannot finish and the transfer complete status bit cannot be set.

When PROCTL[SABGREQ]is set, transfer stops at current block completion on the SD interface and IRQSTAT[BGE] and IRQSTAT[TC] event for the stop request are generated.

When the Continue request is asserted after Stop at block gap, the host starts to transfer remaining blocks. Since the host pre-fetches one extra word from the buffer at stop at block gap, it either transfers corrupted data to the card and thus generates Write CRC status error (IRQSTAT[DCE]) or shuts off the SD clock, which prevents transfer complete(IRQSTAT[TC]) generation and creates deadlock.

This occurs because one extra word is pre-fetched after the block transfer is stopped at block gap and when Continue is requested—the buffer is left with one word less than actual block size (because of one pre-fetch after the previous block).

Impact: Continue cannot be used after Stop at block gap.

Workaround: Software should not set PROCTL[SABGREQ] as this enables stop at block gap request.

Fix plan: No plans to fix

ESDHC 17: Corrupted data read from eSDHC for certain values of WML[RD_WML] and BLKATTR[BLKSIZE]

Description: Corrupted data may be read from the internal eSDHC buffer if previous buffer read access and buffer prefetch occurs at the same time. This is valid for both CPU polling and DMA modes. This issue is observed only when WML[RD_WML] or ((BLKATTR[BLKSIZE] + 3) ÷ 4) is set to an odd value.

Impact: Restriction on using certain values of WML[RD_WML] and BLKATTR[BLKSIZE].

Workaround: BLK_SIZE_IN_WORDS and WML[RD_WML] should not be odd; where BLK_SIZE_IN_WORDS = ((BLKATTR[BLKSIZE] + 3) ÷ 4).

Fix plan: No plans to fix

ESDHC 18: Invalid generation of Block Gap Event

Description: The block gap event (IRQSTAT[BGE]) may be wrongly generated if the stop at block gap request is asserted during the transfer of last block while performing a read or write operation.

Impact: IRQSTAT[BGE] may not be correct when PROCTL[SABGREQ] is set.

Workaround: When IRQSTAT[BGE] is set, check the block count (BLKATTR[BLKCNT]). If it is zero, then ignore the IRQSTAT[BGE] bit.

Fix plan: No plans to fix

ESDHC 19: Invalid IRQSTAT[TC] is set for synchronous abort

Description: An invalid IRQSTAT[TC] is set for a synchronous abort (CMD12 is issued after Stop at Block Gap). According to the device's reference manual, changing CDIHB from 1 to 0 generates a transfer complete interrupt, except in the case when a command with busy is finished. Since CMD12 is also a command with busy, a TC interrupt should not be generated here.

Impact: An invalid IRQSTAT[TC] is set for a synchronous abort. The software should ignore it.

Workaround: None.

Fix plan: No plans to fix

ESDHC 20: BLKATTR[BLKCNT] is not reliable for Block Gap Stop when BLKATTR[BLKZSE] ≤ 4

Description: After IRQSTAT[BGE] and IRQSTAT[TC] are set at block gap, the number of blocks left in BLKATTR[BLKCNT] is one less than the expected value. This happens when the value of BLKATTR[BLKZSE] is less than or equal to 4.

Impact: BLKATTR[BLKCNT] is not reliable for stop at block gap when BLKATTR[BLKZSE] ≤ 4.

Workaround: Only use the value of BLKATTR[BLKZSE] when it is greater than 4 bytes.

Fix plan: No plans to fix

eSDHC-A001: Data timeout counter (SYSCTL[DTOCV]) is not reliable for values of 0x4, 0x8, and 0xC

Description: The data timeout counter (SYSCTL[DTOCV]) is not reliable for DTOCV values 0x4(2^17 SD clock), 0x8(2^21 SD clock), and 0xC(2^25 SD clock). The data timeout counter can count from 2^13–2^27, but for values 2^17, 2^21, and 2^25, the timeout counter counts for only 2^13 SD clocks.

Impact: SYSCTL[DTOCV] is not reliable for values 0x4, 0x8, and 0xC. These values cannot be used.

Workaround: Program one more than the affected value for SYSCTL[DTOCV]. Instead of programming the values of 4, 8, and 12, the SYSTCTL[DTOCV] should be 5, 9, and 13, respectively.

Fix plan: No plans to fix

PMC 1: Cannot wake up from Jog mode

Description: When a jog mode request is initiated by writing 1 to POWMGTCSR[JOG], the core hangs.

Impact: The Jog mode cannot be used due to this bug.

Workaround: Implement the Jog Mode by performing Deep Sleep with frequency change.

1. Write a new clock ratio to PMJCR
2. Configure a timer to wake up as soon as you enter deep sleep. The timer should be the smallest number possible that works
3. Configure PMRCCR[VRCNT_PRE] = 0x01, PMRCCR[VRCNT] = 0x00, PMPDCCR[PDCNT_PRE] = 0x01, PMPDCCR[PDCNT] = 0x00. PMRCCR[RCNT_PRE] and PMRCCR[RCNT] should be set to the minimum value that will give 100 ms PLL lock for the platform frequency in use.
4. Enter deep sleep by writing 1 to POWMGTCSR[DPSLP].

Fix plan: Fixed in Rev 1.1

PMC 2: PMRCCR[RCNT_PRE] register reset value is incorrect

Description: The register PMRCCR[RCNT_PRE] (offset 0xE0084) has a default value of 0x08 in silicon Rev 1.0 instead of the 0x09 on future revisions. The power supply may ramp cycle too quickly, and the chip may not function properly due to this erratum.

Impact: The default setting for PMRCCR[RCNT_PRE] is not large enough to guarantee the minimum 100 µs reset time. Software must overwrite this register to a larger value.

Workaround: Software must write PMRCCR[RCNT_PRE] to 0x09 or a sufficiently large number prior to entering Deep Sleep or performing a Jog Mode request to guarantee a minimum 100 µs reset time.

Fix plan: Fixed in Rev 1.1

JTAG 1: The Debugger cannot read and write to the DDR SDRAM or local bus memory-mapped regions during the deep sleep mode

Description: During the deep sleep mode, DDR and eLBC memory cannot be accessed regardless of the setting of PMCDR[SAP].

Impact: The DDR SDRAM and local bus memory-mapped regions can not be accessed during the deep sleep mode. It limits the usage of customer debug tools only for deep sleep debug.

Workaround: None

Fix plan: Fixed in Rev 1.1

JTAG 2: JTAG debugger cannot access L2 cache as memory-mapped SRAM

Description: Due to this erratum, debug command transactions will not be able to access the L2 cache as memory-mapped SRAM. Debug command transactions to the rest of the system memory map is unaffected.

Impact: JTAG debugger cannot access L2 cache as memory-mapped SRAM.

Workaround: None

Fix plan: Fixed in Rev 1.1

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: MPC8536ECE
Rev. 4
09/2011