P2020 and P2010 Chip Errata

This document details all known silicon errata for the P2020 and P2010. The following table provides a revision history for this document.

Table 1. Document revision history

Revision	Date	Significant changes
CC	01/2016	Added eSDHC A-009204 Modified eTSEC A-006514
BB	03/2015	Added the following errata: • eSDHC A-008358 • MPIC A-008312 Modified Ethernet A-007207 Renamed eTSEC 11 to A-006514 Re-ordered the eTSEC and USB sections numerically
AA	05/2014	Added the following errata: • CPU A-006022 • eTSEC A-006293, A-007734 • Ethernet A-007207 • USB 10, A-003829, A-003832, A-003834, A-003838, A-003840, A-003842, A-003843, A-003844, A-003845, A-003846, A-003848, A-003849, A-003850, A-004477, A-005375, A-005697 Modified • CPU A-005125, A-006184 • eTSEC 9, A-006502 • USB 2 Renamed eTSEC-A002 to A-006502 Removed the following errata: • eTSEC 18 • USB A-003817
Z	03/3013	Added CPU erratum A-006184 Modified erratum: • CPU A-005125 • USB A-003827

Table continues on the next page...

Table 1. Document revision history (continued)

Revision	Date	Significant changes
		Re-ordered errata alphabetically by group
Y	11/2012	Added eLBC erratum A-004893; eSDHC errata ESDHC 20, A-004577, A-005055; USB errata A-003827, A-003837. Modified CPU A-004801
X	7/2012	Renamed DUART 1 to A-004737 and renamed USB 9 to USB erratum A-003817. Added CPU erratum A-005125 and DDR erratum A-004508.
W	2/2012	Added CPU erratum A-003477, USB erratum A003836, and eSDHC erratum A-004373. Updated PCI-Ex 1 fix plan. Modified CPU 2 and DDR 2 workaround and USB-A001.
V	5/2011	Added USB-A005 and USB-A007 Modified impact for SEC-A001 Modified workaround for USB 3
U	11/2010	In Table 2, added silicon revision 2.1 SVR and PVR information In Table 3, added silicon revision 2.1 column Updated "Fix Plan" for CPU-A005, DMA-A001, GEN-A007, IEEE 1588-A001 Modified CPU 1 impact
T	10/2010	Added CPU-A005, SEC-A001, and SRIO-A004 Updated fix plan for eTSEC23 Modified USB-A002 and USB-A003
S	08/2010	Added GEN-A007, SRIO-A002, and USB-A003 Modified workaround for eSDHC 12 Updated fix plan for DDR2 Removed eTSEC7 and eTSEC21
R	06/2010	Modified eTSEC-A002 and IEEE1588-A001 Modified Fix plan for DMA-A001
Q	05/2010	Modified workaround for DMA-A001 and DDR2 Updated Fix plan for DDR 7 to "No plans to fix"
P	04/2010	Updated Fix plan for DDR 1 to "Fixed in Rev. 2.0" Added GEN-A004
O	—	Skip Rev. O
N	03/2010	Changed fix plan (projected solution) for ESDHC 5, and PIC 1, to "Fixed in Rev 2.0." Changed fix plan (projected solution) for eTSEC 3 to "No plans to fix" Changed fix plan of other errata which had been "To be fixed in Rev 2.0" to "Fixed in Rev 2.0." Updated CPU 3 and eTSEC 20 descriptions Updated eSDHC13 description Modified eSDHC-A001 workaround Added DMA-A001, eLBC-A001, eTSEC-A002, PCIe-A001, and IEEE1588-A001 In Table 2, added silicon revision 2 column

Table continues on the next page...

Table 1. Document revision history (continued)

Revision	Date	Significant changes
M	9/2009	Updated eTSEC15 impact typo, Added USB-A001, USB-A002 Added eSDHC-A001 Added eTSEC-A001
L	8/2009	Added DDR8, eSDHC19, PCI-Ex 1, eTSEC23
K	7/2009	Update DDR2 workaround (step d) Added DDR7

J	6/2009	Update DDR3, DDR6
		Added USB8, USB9
		Added eTSEC20, eTSEC21, eTSEC22
		Added CPU6
		Added eSDHC17, eSDHC18

I	—	Skip Rev. I
H	5/2009	Added USB5-7. Added eSDHC16 Added CPU5 Added eSDHC19
G	4/2009	Update eTSEC2 ESDHC15
F	04/2009	Added DDR6, USB3-4, CPU4, ESDHC12-14 Update DDR4 Corrections in Table 2 PVR
E	02/2009	Added DDR 5, SRIO 3, and ESDHC 12.
D	01/2009	Added PIC 1 (P2010 only), SEC1, eTSEC16-18, and DDR3-4.
C	11/2008	Removed CPU 3. Added PM1, CPU3, and SRIO2.
B	11/2008	Removed PM1 Modified Projected Solution for the following errata: DDR2, eLBC, GEN2, eTSEC6, eTSEC9, eTSEC10, eTSEC11. eTSEC12, I2C1 and USB2
A	10/2008	Initial NDA release.

The following table 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	Note
P2020	2.1	5.1	0x8021_1051	0x80EA_0021	With Security
P2020	2.1	5.1	0x8021_1051	0x80E2_0021	Without security
P2010	2.1	5.1	0x8021_1051	0x80EB_0021	With Security
P2010	2.1	5.1	0x8021_1051	0x80E3_0021	Without security
P2020	2.0	5.0	0x8021_1050	0x80EA_0020	With Security
P2020	2.0	5.0	0x8021_1050	0x80E2_0020	Without security
P2010	2.0	5.0	0x8021_1050	0x80EB_0020	With Security
P2010	2.0	5.0	0x8021_1050	0x80E3_0020	Without security
P2020	1.0	4.0	0x8021_1040	0x80EA_0010	With Security
P2020	1.0	4.0	0x8021_1040	0x80E2_0010	Without security
P2010	1.0	4.0	0x8021_1040	0x80EB_0010	With Security
P2010	1.0	4.0	0x8021_1040	0x80E3_0010	Without security

Table 3 summarizes all known errata and lists the corresponding silicon revision level to which they apply. A ‘Yes’ entry indicates the erratum applies to a particular revision level, and an ‘—’ entry means it does not apply.

Table 3. Summary of Silicon Errata and Applicable Revision

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	2.0	2.1
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	e500 core signal glitch during reset may cause the device to behave erratically	Fixed in Rev 2.0	Yes	No	No
CPU 4	Increased latency on coherent transactions when executing a block of back-to-back ICBI instructions	Fixed in Rev 2.0	Yes	No	No
CPU 5	TLB flash invalidates must be tightly controlled by software	Fixed in Rev 2.0	Yes	No	No
CPU 6	Core operating frequency may exceed the frequency established at Power-on Reset for the core PLL ratio	Fixed in Rev 2.0	Yes	No	No
A-001428	Enabling IEEE 754 exceptions can cause errors	Fixed in Rev 2.1	Yes	Yes	No
A-003477	Phantom branches in the BTB may not be correctly invalidated	No plans to fix	Yes	Yes	Yes
A-005125	In a very rare condition, a system hang is possible when the e500 core initiates a guarded load to PCI/PCIe/SRIO while the PCI/PCIe/SRIO performs a coherent write to memory	No plans to fix	Yes	Yes	Yes
A-006022	PMGC0 and UPMGC0[TBSEL and TBEE] can not be read	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	2.0	2.1
A-006184	Simultaneous Instruction L1 MMU (I-L1VSP) miss (due to eviction) and interrupt servicing can cause a core hang	No plans to fix	Yes	Yes	Yes
DDR					
DDR 1	If a single address parity error is detected, the ERR_DETECT[MME] bit will be falsely set	Fixed in Rev 2.0	Yes	Yes	Yes
DDR 2	DDR controller may not work with dual-ranked DDR3 registered DIMM if write leveling is enabled	Fixed in Rev 2.0	Yes	No	No
DDR 3	Asserting panic interrupt via $\overline{\text{IRQ\_OUT}}$ the DDR controller may violate the JEDEC specifications	Fixed in Rev 2.0	Yes	No	No
DDR 4	DDR to CCB clock ratios of greater than 1.7:1 can result in DDR configuration register corruption	Fixed in Rev 2.0	Yes	No	No
DDR 5	The DDR controller may fail the write leveling sequence for DDR3	Fixed in Rev 2.0	Yes	No	No
DDR 6	Data may become corrupted for DDR3 32-bit bus mode	Fixed in Rev 2.0	Yes	No	No
DDR 7	DDR I/Os may not meet JEDEC speced VOH, VOL voltages	No plans to fix	Yes	Yes	Yes
DDR 8	Memory controller perfmon counter for read beats is inaccurate	No plans to fix	Yes	Yes	Yes
A-004508	DDR controller may not function across the full industrial temperature range	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	Fixed in Rev 2.1	Yes	Yes	No
DUART					
A-004737	BREAK detection triggered multiple times for a single break assertion	No plans to fix	Yes	Yes	Yes
eLBC					
eLBC 1	LTEATR and LTEAR may show incorrect values under certain scenarios	No plans to fix	Yes	Yes	Yes
eLBC 2	Multi-bank DRAM and SDRAM do not work without any external logic	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
eSDHC					
eSDHC 1	Force Event Register	No plans to fix	Yes	Yes	Yes
eSDHC 2	eSDHC indicates AUTO CMD12 interrupt later than expected	No plans to fix	Yes	Yes	Yes
eSDHC 3	Data corruption during write with pause operation	No plans to fix	Yes	Yes	Yes
eSDHC 4	Incorrect data is written to a card after transfer paused	No plans to fix	Yes	Yes	Yes
eSDHC 5	CRC might be corrupted for write data transfer if it is preceded by read transfer	Fixed in Rev 2.0	Yes	No	No
eSDHC 6	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

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	2.0	2.1
eSDHC 7	During multi-write operation, unexpected Transfer Complete (IRQSTAT[TC] register) bit can be set	No plans to fix	Yes	Yes	Yes
eSDHC 8	eSDHC interrupt not negated in case of card insertion/ removal	No plans to fix	Yes	Yes	Yes
eSDHC 9	Unable to issue CMD12 if SD clock shuts off due to slow system access	No plans to fix	Yes	Yes	Yes
eSDHC 10	Manual Asynchronous CMD12 abort operation causes protocol violations	No plans to fix	Yes	Yes	Yes
eSDHC 11	PRSSTAT[CIDHB] is not reliable for commands with busy (R1b)	No plans to fix	Yes	Yes	Yes
eSDHC 12	eSDHC cannot finish write operation after continuing from Block Gap Stop	No plans to fix	Yes	Yes	Yes
eSDHC 13	Corrupted data read from eSDHC for certain values of WML[RD_WML] and BLKATTR[BLKSIZE]	No plans to fix	Yes	Yes	Yes
eSDHC 14	Invalid generation of Block Gap Event	No plans to fix	Yes	Yes	Yes
ESDHC 15	eSDHC DMA and interrupt not functional	Fixed in Rev 2.0	Yes	No	No
eSDHC 16	BLKATTR[BLKCNT] is not reliable for Block Gap Stop when BLKATTR[BLKZSE] ≤ 4	No plans to fix	Yes	Yes	Yes
eSDHC 17	Invalid IRQSTAT[TC] is set for synchronous abort	No plans to fix	Yes	Yes	Yes
ESDHC 18	The HOSTCAPBLT register does not report to host driver correct value for eSDHC interface operating voltage supported	Fixed in Rev 2.0	Yes	No	No
ESDHC 19	DSADDR register status is not reliable	No plans to fix	Yes	Yes	Yes
eSDHC 20	CMD CRC error or CMD index error may be set for CMD without data while CMD with data is in progress	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
A-004373	Host may not detect SD card insertion/removal using DAT3 signal	No plans to fix	Yes	Yes	Yes
A-004577	PRSSTAT[DLA] bit does not reflect the data line state when any command with busy (R1b) is issued	No plans to fix	Yes	Yes	Yes
A-005055	A glitch is generated on the card clock with software reset or a clock divider change	No plans to fix	Yes	Yes	Yes
A-008358	Data timeout error may occur earlier then expected	No plans to fix	Yes	Yes	Yes
A-009204	System bus may hang if software register SYSCTL[RSTD] is set while DMA transactions are active	No plans to fix	Yes	Yes	Yes
Ethernet					
A-007207	TBI link status bit may stay up after SGMII electrical idle is detected	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	Fixed in Rev 2.0	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	2.0	2.1
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
eTSEC 8	Half-duplex collision on FCS of Short Frame may cause Tx lockup	No plans to fix	Yes	Yes	Yes
eTSEC 9	Ethernet controller does not exit from Magic Packet mode when an oddly formed Magic Packet is received	No plans to fix	Yes	Yes	Yes
eTSEC 10	MAC: Malformed Magic Packet Triggers Magic Packet Exit	Fixed in Rev 2.0	Yes	No	No
eTSEC 12	Receive pause frame with PTV = 0 does not resume transmission	Fixed in Rev 2.0	Yes	No	No
eTSEC 13	Odd prescale values not supported	No plans to fix	Yes	Yes	Yes
eTSEC 14	IEEE 1588 accuracy can be adversely impacted in systems using multiple unsynchronized gigabit Ethernet ports	Fixed in Rev 2.0	Yes	No	No
eTSEC 15	TxBD polling loop latency is 1024 bit-times instead of 512	No plans to fix	Yes	Yes	Yes
eTSEC 16	Misfiled Packets Due to Incorrect Rx Filer Set Mask Rollback	Fixed in Rev. 2.0	Yes	No	No
eTSEC 17	1588 alarm fires when programmed to less than current time	No plans to fix	Yes	Yes	Yes
eTSEC 19	Read of MIB counters may return zeros if ECNTRL[AUTOZ]=1	Fixed in Rev 2.0	Yes	No	No
eTSEC 20	Excess delays when transmitting TOE=1 large frames	Fixed in Rev 2.0	Yes	No	No
eTSEC 22	Data corruption may occur in SGMII mode	Fixed in Rev 2.0	Yes	No	No
eTSEC 23	Controller may not be able to transmit pause frame during pause state	Fixed in Rev 2.0	Yes	No	No
eTSEC-A001	MAC: Pause time may be shorter than specified if transmit in progress	Fixed in Rev 2.0	Yes	No	No
A-006293	Mixing TOE = 0 and TOE = 1 frames may cause data corruption	No plans to fix	Yes	Yes	Yes
A-006502	Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame	No plans to fix	Yes	Yes	Yes
A-006514	No parser error for packets containing invalid IPv6 routing header packet	No plans to fix	Yes	Yes	Yes
A-007734	Stale time stamps and over-flow conditions can occur when using an external 1588 input clock at certain frequencies	No plans to fix	Yes	Yes	Yes
GEN					
GEN 1	eLBC PLL has incorrect default VCO selection	Fixed in Rev 2.0	Yes	No	No
GEN 2	Non-functional performance monitor events for ECM to DDR transactions.	No plans to fix	Yes	Yes	Yes
GEN-A004	READY_P1, READY_P0 Functionality	Fixed in Rev. 2.0	Yes	No	No
GEN-A007	Momentary glitch on I/Os during HRESET assertion and de-assertion	Fixed in Rev 2.1	Yes	Yes	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	2.0	2.1
I2C					
I2C 1	Enabling I^2C could cause I^2C bus freeze when other I^2C devices communicate	No plans to fix	Yes	Yes	Yes
IEEE1588					
IEEE1588-A001	Incorrect received timestamp or dropped packet when 1588 time-stamping is enabled	Fixed in Rev 2.1	Yes	Yes	No
MPIC					
A-008312	Duplicate edge-triggered interrupt after priority rearbitration	No plans to fix	Yes	Yes	Yes
PCIe					
PCI-Ex 1	Excess correctable errors in receiving DLLPs and TLPs on x2 link	Fixed in Rev 2.0	Yes	No	No
PCIe-A001	PCI Express Hot Reset event may cause data corruption	Fixed in Rev 2.0	Yes	No	No
PIC					
PIC 1	NCPU of Feature reporting register does not reflect number of cores supported by device (P2010 only).	Fixed in Rev 2.0	Yes	No	No
PM					
PM 1	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
SEC					
SEC 1	STHA (Snow 3G Execution Unit) not reflected in the Controller EU Assignment Status Register (EUASR)	No plans to fix	Yes	Yes	Yes
SEC-A001	Channel Hang with Zero Length Data	No plans to fix	Yes	Yes	Yes
sRIO					
SRIO 1	CCSR[PW] does not reflect hardware width of port	No plans to fix	Yes	Yes	Yes
SRIO 2	Serial RapidIO Packets with errors are not ignored by the controller while in input-retry-stopped state	Fixed in Rev 2.0	Yes	No	No
SRIO 3	Message unit cannot generate messages with priority 0	No plans to fix	Yes	Yes	Yes
SRIO-A002	SRIO reset command does not result in device reset	No plans to fix	Yes	Yes	Yes
SRIO-A004	SRIO controller may incorrectly transmit or block responses when PmCCSR[OPE] = 0	No plans to fix	Yes	Yes	Yes
USB					
USB 1	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 2	SE0_NAK issue	Fixed in Rev 2.0	Yes	No	No
USB 3	Missing SOFs and false babble error due to Rx FIFO overflow	No plans to fix	Yes	Yes	Yes
USB 4	No error interrupt and no status will be generated due to ISO mult3 fulfillment error	No plans to fix	Yes	Yes	Yes
USB 5	Wrong value read in BURSTSIZE[TXPBURST] and BURSTSIZE[RXPBURST] registers	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	2.0	2.1
USB 6	Core device fails when it receives two OUT transactions in a short time	No plans to fix	Yes	Yes	Yes
USB 7	CRC not inverted when host under-runs on OUT transactions	No plans to fix	Yes	Yes	Yes
USB 8	NAK counter decremented after receiving a NYET from device	No plans to fix	Yes	Yes	Yes
USB 10	Read of PERIODICLISTBASE after successive writes may return a wrong value in host mode	No plans to fix	Yes	Yes	Yes
USB-A001	Multiple dTDs can cause USB device controller unprimed an end point	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 TX CMD 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
A-003827	DATA PID error interrupt issued twice for the same high bandwidth ISO transfer	No plans to fix	Yes	Yes	Yes
A-003829	Host detects frame babble but does not halt the port or generate an interrupt	No plans to fix	Yes	Yes	Yes
A-003832	Device NAKs OUT transaction if the host misses a handshake and retries	No plans to fix	Yes	Yes	Yes
A-003834	Host ACKs data2 PID sent by bulk endpoint when it should send BTO	No plans to fix	Yes	Yes	Yes
A-003836	Host does not retire ISO transfer if the first or second packet in MULT sequence is short.	No plans to fix	Yes	Yes	Yes
A-003837	When operating in test mode, the CSC bit does not get set to 1 to indicate a change on CCS	No plans to fix	Yes	Yes	Yes
A-003838	Device ACKs a DATA1 PID after SETUP token	No plans to fix	Yes	Yes	Yes
A-003840	The CERR is not decremented, and the xact err bit is not updated on a NYET handshake to a SETUP	No plans to fix	Yes	Yes	Yes
A-003842	Device controller may count below 3ms when detecting a Suspend state	No plans to fix	Yes	Yes	Yes
A-003843	Non-double word aligned buffer address sometimes causes host to hang on OUT retry	No plans to fix	Yes	Yes	Yes
A-003844	Host does not retire iTD but issues remaining transaction in next uframe	No plans to fix	Yes	Yes	Yes
A-003845	Frame scheduling robustness-Host may issue token too close to uframe boundary	No plans to fix	Yes	Yes	Yes
A-003846	Host does not pre-fill TxFIFO correctly when data buffer address is not word aligned	No plans to fix	Yes	Yes	Yes
A-003848	ISO IN - dTD not retired if MULT field is not correctly 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	2.0	2.1
A-003849	USB FORCE_ENABLE_LS feature not supported	No plans to fix	Yes	Yes	Yes
A-003850	A write operation to PERIODICLISTBASE hangs system bus if PHY is in low power mode	No plans to fix	Yes	Yes	Yes
A-004477	ULPI Function Control Register write gets corrupted by a soft reset	No plans to fix	Yes	Yes	Yes
A-005375	Full speed 'J' driven in host mode while doing remote wakeup	No plans to fix	Yes	Yes	Yes
A-005697	Suspend bit asserted before the port is in Suspend state	No plans to fix	Yes	Yes	Yes

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: e500 core signal glitch during reset may cause the device to behave erratically

Description: After negation of $\overline{\text{HRESET}}$, some of the internal signals between the e500 core and the platform may glitch. If the signals glitch, various erratic behaviors can happen. Two known issues are that the device may enter check-stop state and hang upon wake up from reset. The device may also re-enter the reset state.

This failure is a function of either the core supply voltage, temperature platform clock frequency, platform-to-core clock ratio, the wafer processing parametric, or a combination of the above.

Impact: If the glitch occurs, the device may fail to boot or fail to function properly.

Workaround: To avoid these failing scenarios, the internal registers or flip-flops in the e500 core need to be preconditioned. Use one of the following workarounds.

- Implement a workaround circuit that presets the internal latch during reset. A circuit diagram and code to implement a workaround is available from Freescale.
- Use Freescale CodeWarrior or a third-party partner's emulator/debugger that presets the internal latch, which avoids the errata.

Fix plan: Fixed in Rev 2.0

CPU 4: Increased latency on coherent transactions when executing a block of back-to-back ICBI instructions

Description: While the e500 core is executing a block of back-to-back ICBI instructions, it may be unable to respond to coherent transactions. Any transactions that are marked as coherency-required may be effectively stalled until completion of the ICBI instructions. For example, assume that another core executes a load or store instruction to a page that is marked coherency required, and a transaction is required to fetch the cache line for that load or store (that is, the line is not present in the core's L1 cache). The coherent transaction may be stalled by the block of ICBI instructions, which may stall the processor performing the load or store. The same scenario can occur for I/O devices performing coherent transactions.

Impact: Back-to-back ICBI instructions are often used to clear a page of memory when it is being allocated as an instruction page. For example, a block of 128 ICBI instructions are required to clear a 4-Kbyte instruction page. During this time, other processors and I/O devices (for example, PCI-Express) performing coherent transactions may see long latencies.

Workaround: Break large blocks of ICBI instructions into smaller blocks separated by MSYNCs. For example, execute an MSYNC after every 2, 4, 8, etc. ICBI instructions. This gives other cores and I/O devices an opportunity to perform their coherent transactions.

Fix plan: Fixed in Rev 2.0

CPU 5: TLB flash invalidates must be tightly controlled by software

Description: This erratum refers to any flash invalidate of TLB0 or TLB1, including the following:

- Writing 1 to MMUCSR0[TLB0_FI]
- Writing 1 to MMUCSR0[TLB1_FI]
- Executing **tlbivax** with EA[61] set to 1

If this is not tightly controlled by software, it may yield incorrect results.

Impact: Execution of TLB flash invalidate may yield an undesirable effect such as a data fetch from a wrong address as well as execution pointer getting corrupted.

Workaround: Flash invalidating TLB0 or TLB1 should be avoided. **tlbivax** instructions that do not set EA[61] should be used to perform invalidations, or **tlbwe** can be used to clear the valid bit (by setting MAS1[V] to 0 prior to executing **tlbwe**).

If the TLB must be completely invalidated (as would occur for a Flash invalidate operation), because software does not know the contents of the TLB, the following algorithm can be used in place of the Flash invalidate.

For TLB0:

```
MAS1[V] = 0; MAS0[TLBSEL] = 0;
for index  = 0 to 127      // for each set
{
  MAS2[EPN] = index;
  for esel = 0 to 3        // for each way
  {
    MAS0[ESEL] = esel;
    tlbwe;                 // invalidate entry
  }
}
isync;
```

TLB0 should only be invalidated using a series of TLB writes.

For TLB1:

```
// TLB 1 needs more care, because IPROT entries should not be invalidated
  MAS0[TLBSEL] = 1;
  for esel  = 0 to 15          // for each entry
  {
    MAS0[ESEL] = esel;
    tlbre;
    if MAS1[V] == 1 then    // only invalidate if already valid
    {
      if MAS1[IPROT] == 0 then     // only invalidate non-IPROT entries
      {
        MAS1[V] = 0;
        tlbwe;             // invalidate entry
      }
    }
  }
  isync;
```

Fix plan: Fixed in Rev 2.0

CPU 6: Core operating frequency may exceed the frequency established at Power-on Reset for the core PLL ratio

Description: When the core (cfg_core_speed) configuration input is configured for a frequency greater than 1000 MHz at power-on-reset configuration, the core PLL may speed-up to a frequency that exceeds the maximum frequency of the core. The probability of this erratum occurring increases as the operating junction temperature is increased.

Impact: This erratum results in the core functional failure.

Workaround: For core operation above 1000 MHz, the following configuration change is needed.

- For core 0 configuration (cfg_core0_speed), LA24 pin must be pulled low (logic 0).
- For core 1 configuration (cfg_core1_speed), LA25 pin must be pulled low (logic 0).

For core operation at or below 1000 MHz, this erratum does not apply.

Fix plan: Fixed in Rev 2.0

A-001428: 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 unrelated floating point instruction that has been cancelled on the mispredicted 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: Fixed in Rev 2.1

A-003477: Phantom branches in the BTB may not be correctly invalidated

Affects: CPU

Description: The branch target buffer (BTB) holds effective addresses associated with a branch instruction. A process context switch might bring in another task whose MMU translations are such that it uses the same effective address for another nonbranch instruction for which the BTB has an entry for a previously encountered branch. This causes the fetch unit to redirect instruction fetch to the BTB's target address. When this occurs it is called a phantom branch. Later, during execution of the instruction, the hardware realizes the error and is supposed to evict the BTB entry.

However, with this erratum, a BTB entry of a phantom branch may not be invalidated when the phantom branch is decoded from instruction buffer 0, and

1. the BTB hit a phantom branch and the branch address does not equal the fetch group address, that is, the branch is not the first instruction in the fetch group, OR,
2. the BTB hit a phantom branch and the branch address equals the fetch group address.

In case 1, where the fetch group address and branch address are not equal, the BTB will not be invalidated. In case 2, after two attempts to issue the phantom branch, the BTB entry will be properly invalidated. In both cases, it is possible that a valid entry will be invalidated instead.

Impact: Performance may be impacted due to possible additional phantom branches. This errata could occur when switching across processes that share the same effective address space.

Workaround: Select one of the following options, depending on which results in the best performance:

- Continue to use the BTB as it currently operates.
- Invalidate the BTB (BUCSR [BBFI] =1) at the appropriate points to ensure a phantom branch never occurs. (Possible scenarios that can cause phantom branches include, but are not limited to, the following: switching contexts where an exception handler address space overlaps with user code space, while running self-modifying code, 64-bit programs executing across 4G segments, during any process switch, and so on.)
- Disable the BTB (BUCSR[BPEN] = 0) temporarily without invalidating the BTB when switching to other contexts that may cause phantom branches. Re-enable the BTB when switching back to the main context. This allows the BTB contents to remain intact for the main context such that when returning back to the main context, the BTB is valid.
- Disable BTB (BUCSR[BPEN] = 0) completely for all contexts.

Each workaround impacts performance depending on the application.

Fix plan: No plans to fix

A-005125: In a very rare condition, a system hang is possible when the e500 core initiates a guarded load to PCI/PCIe/SRIO while the PCI/PCIe/SRIO performs a coherent write to memory

Affects: CPU

Description: When the e500 core initiates a guarded load to the PCI/PCIe/SRIO while the PCI/PCIe/SRIO performs a write to cacheable, coherent memory, and that write is behind (or is one of) multiple full cache line writes from an IO device that hit modified in the e500's L1 data cache, it is possible for the CCB bus arbiter to enter into an invalid state and hang the system. A very specific sequence of streamed IO snoops and retries from a congested memory system must occur just before the PCI/PCIe/SRIO write reaches the core for the hang to occur.

Impact: When the erratum is encountered, no further forward progress is made to and from the e500 coherency module (ECM), and the system may hang.

Workaround: Set SPR976[40:41] to b'10. Setting these bits avoids the hang condition by forcing the core to process all snoops of IO device full cache line writes to DDR differently. This setting does not impact performance.

Fix plan: No plans to fix

A-006022: PMGC0 and UPMGC0[TBSEL and TBEE] can not be read

Affects: CPU

Description: PMGC0[TBSEL and TBEE] can be correctly written but will always return 0 when read. UPMGC0[TBSEL and TBEE] will also always contain 0. No other PMGC0 and UPMGC0 bits are impacted by this erratum.

Impact: Software can correctly write PMGC0[TBSEL and TBEE] and they will function as specified. However, software must remember what values it wrote and not rely on the value it reads from PMGC0 or UPMGC0[TBSEL and TBEE].

Workaround: None

Fix plan: No plans to fix

A-006184: Simultaneous Instruction L1 MMU (I-L1VSP) miss (due to eviction) and interrupt servicing can cause a core hang

Affects: CPU

Description: A system hang is possible when an exception occurs at the same time as several other internal core conditions occur. For the hang to happen, the L2 MMU TLB1 entry, which maps the translation for the first instruction in the exception handler, must not be in the I-L1VSP.

Impact: No further forward progress will be made until a higher priority exception is received for which interrupts are enabled.

Workaround: Option 1 (All of the following steps must be performed):

1. Do not use more than 4 TLB1 entries, and ensure all interrupt handlers are mapped to one of the 4 TLB1 entries with a TID=0 and IPROT=1.
2. Prevent explicit invalidation of the TLB1 entry that contains the interrupt handler page by not overwriting or invalidating it (never clear the valid bit).
3. Prevent non-explicit invalidations of the TLB1 entry that contains the interrupt handler page by never:
   a. Setting MMUCSR0[L2TLB1_FI].
   b. Executing a **tlbivax** with RA[60:61] set to 0b11 (invalidate all).

   This should not be executed from any core in the integrated device as **tlbivax** is broadcast to all cores.

Software may perform non-explicit invalidations if:

- The instructions for invalidation are mapped by the same TLB1 entry which maps the interrupt handlers. This will normally be the case if the operating system maps all its static executable code with a single TLB1 entry; AND
- All interrupts should be blocked while performing the actions (set MSR[EE], MSR[CE], MSR[ME] and MSR[DE] = 0).

Option 2 (All of the following steps must be performed):

1. Do not use more than 4 Instruction TLB1 entries and ensure all interrupt handlers are mapped to one of the 4 TLB1 entries with a TID = 0 and IPROT = 1.
2. Ensure fetch address consistency by:
   a. Ensuring data and instruction pages are not contiguous.
   b. Ensuring virtual addresses used for instructions and data do not overlap.
   c. Never remapping virtual addresses between instruction and data.
3. Prevent explicit invalidation of the TLB1 entry that contains the interrupt handler page by not overwriting or invalidating it (never clear the valid bit).
4. Prevent non-explicit invalidators of the TLB1 entry that contains the interrupt handler page by never:
   a. Setting MMUCSR0[L2TLB_F1].
   b. Executing a **tlbivax** with RA[60:61] set to ob11 (invalidate all). This should not be executed from any core in the integrated device as **tlbivax** is broadcast to all cores.

Software may do non-explicit invalidations if:

- The instructions for invalidation are mapped by the same TLB1 entry which maps the interrupt handlers. This will normally be the case if the operating system maps all its static executable code with a single TLB1 entry; AND
- All interrupts should be blocked while performing the actions (set MSR[EE], MSR[CE], MSR[ME], and MSR[DE] = 0).

Option 3 (All of the following steps must be performed):

1. In order to restart a core that has hung due to this erratum, set the watchdog timer to take an interrupt at some period greater than the decrementer interrupt interval, but smaller than an unacceptable hang time.
2. Have software reset the watchdog timer trigger during the decrementer interrupt, ensuring the core only takes the watchdog interrupt if it is hung.

Notes:

- This will still allow for hangs, but will put a limit on the degradation in performance.
- If the software already performs critical interrupts, the core may still hang. If the customer wants to set the “watchdog timer reset” field (TCR[WRC], (actual value will be SOC dependent), then a core reboot will exit the hang condition.

Fix plan: No plans to fix

DDR 1: If a single address parity error is detected, the ERR_DETECT[MME] bit will be falsely set

Description: The DDR controller should only set ERR_DETECT[MME] if multiple errors of the same type have been detected. However, if ERR_DETECT[APE] is set for an address parity error, then ERR_DETECT[MME] will be falsely set the subsequent cycle.

Impact: The impact is low. If ERR_DETECT[APE] is set due to an address parity error, then customers should also expect ERR_DETECT[MME] to be set. This prevents the controller from detecting multiple address parity errors.

Workaround: None

Fix plan: Fixed in Rev 2.0

DDR 2: 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 (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 DDR3 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 DDR3, memory size = 1 Gbyte.

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

  f. Clear DDR_SDRAM_CFG[MEM_HALT].

Fix plan: Fixed in Rev 2.0

DDR 3: Asserting panic interrupt via 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: Fixed in Rev 2.0

DDR 4: DDR to CCB clock ratios of greater than 1.7:1 can result in DDR configuration register corruption

Description: If the clock ratio between the DDR and CCB is greater than 1.7:1, 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 1.7:1. For example, if a DDR data rate of 667 MHz is used, the minimum CCB frequency should be 392 MHz.

Fix plan: Fixed in Rev 2.0

DDR 5: 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: Fixed in Rev 2.0

DDR 6: 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: Fixed in Rev 2.0

DDR 7: DDR I/Os may not meet JEDEC speced VOH, VOL voltages

Description: For DDR2, the SSTL_18 specifications state that with a 13.4-mA load current, the driver should maintain no more than 0.28 V at the pin, while with a –13.4-mA load current, the driver should maintain no less than 1.42 V on the pin (assuming a 1.7-V supply). These described IOH/IOL with their corresponding VOH/VOL may not be met on DDR I/Os.

Impact: No functional impact is expected from not meeting the IOH/IOL and VOH/VOL on DDR I/Os.

Workaround: None

Fix plan: No plans to fix

DDR 8: Memory controller perfmon counter for read beats is inaccurate

Description: There is a memory controller performance monitor event counter register that counts the total number of read beats on the DRAM interface. This event counter register may not accurately represent the total number of read beats transferred on the DRAM interface.

If a 64-bit interface is used, the count will be as expected. If a 32-bit DRAM interface is used, the count will be half as much as it should. If a 16-bit DRAM interface is used, the count will be 1/4 as much as it should.

Impact: The results of this perfmon event counter will be inaccurate.

Workaround: Multiply the final count by 2 if a 32-bit DRAM data bus is used. Multiply the final count by 4 if a 16-bit DRAM data bus is used.

Fix plan: No plans to fix

A-004508: DDR controller may not function across the full industrial temperature range

Affects: DDR

Description: When the DDR controller is initialized below a junction temperature of 0°C and then operated above a junction temperature of 65°C, the DDR controller may cause receive data errors, resulting in single-bit ECC errors, multi-bit ECC errors and/or corrupted data.

Impact: Data corruption may be observed during reads from DRAM.

Workaround: When the DDR controller is initialized below a junction temperature of 0°C and then operated above a junction temperature of 65°C without a reset, then software should set bit 22 at the CCSR register offset 0x0_2F08 before the DDR controller is enabled. This ensures the DDR controller operates across the full, supported industrial temperature range.

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 8 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 ≤ 2 Kbytes. Byte count is set in BCRn[BC].
  - Aligned start addresses, such that each descriptor yields no more than 16 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.

Fix plan: Fixed in Rev 2.1

A-004737: BREAK detection triggered multiple times for a single break assertion

Affects: DUART

Description: Previously DUART 1

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 workaround applies to both polling and interrupt-driven implementations.

Fix plan: No plans to fix

eLBC 1: 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 2: 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-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

eSDHC 1: 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 2: 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 3: 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 4: 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 5: 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 2.0

eSDHC 6: 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 7: 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 8: 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 9: 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 10: 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 when AUTO12EN is also set. Instead, use a (software) synchronous CMD12 or AUTOCMD12 to abort data transfer.

Due to erratum A-004577, CMD13 needs to be sent after TC of the data transfer command for which AutoCMD12 is enabled. See A-004577 for details.

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 11: 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 12: 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 13: 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 14: 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 15: eSDHC DMA and interrupt not functional

Description: After initializing the eSDHC and programming DMA transfer attributes, the DMA does not initiate a data transfer when enabled by setting XFERTYPE[DMAEN]. The eSDHC interrupt signal to the interrupt controller is non-functional and will not assert even if enabled and interrupt conditions exist.

Impact: Due to this bug, boot via the eSDHC interface is not possible. The bug

causes the eSDHC DMA and interrupt are unusable after boot as well.

Workaround: In order to avoid this errata, A JTAG sequence can be applied via JTAG at reset to patch the internal hardware, which fully corrects this bug.

There are two possible workarounds.

Option 1:

Implement a workaround circuit that will pre-set internal latch during reset. A circuit diagram and code to implement a workaround is available from Freescale.

Option 2:

Use Freescale CodeWarrior or Third-party partner's emulator/debugger that will pre-set the internal latch to avoid the errata.

Fix plan: Fixed in Rev 2.0

eSDHC 16: 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 17: 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 18: The HOSTCAPBLT register does not report to host driver correct value for eSDHC interface operating voltage supported

Description: The HOSTCAPBLT register provides software visibility into the operating voltages supported specific to eSDHC implantation. The HOSTCAPBLT[VS18] , HOSTCAPBLT [VS30] when read by software should return value 2‘b10. However, as result of this errata value 2‘b01 is returned when these bits are read by software.

Impact: The HOSTCAPBLT register provides software visibility into the operating voltages supported specific to eSDHC implantation. The HOSTCAPBLT[VS18] , HOSTCAPBLT [VS30] when read by software should return value 2‘b10. However, as result of this errata value 2‘b01 is returned when these bits are read by software.

Workaround: None

Fix plan: Fixed in Rev 2.0

eSDHC 19: DSADDR register status is not reliable

Description: The DMA system address register (DSADDR) is not reliable after end of DMA transfer under certain values of Block Size and Watermark Level. However, data is transferred without any problem.

After the end of a DMA transfer, the DSADDR should point to the system address of the next contiguous data position. However, it sometimes points to one value less than the expected.

Impact: The DSADDR register cannot be used after end of a DMA transfer.

Workaround: None. Software needs to reset the DSADDR every time.

Fix plan: No plans to fix

eSDHC 20: CMD CRC error or CMD index error may be set for CMD without data while CMD with data is in progress

Description: While a command with data is in progress and a command without data is issued, for example CMD13, an invalid command CRC error (IRQSTAT[CCE]) and/or command index error (IRQSTAT[CIE]) might be detected. This can happen when SDHC_CLK shuts off (due to buffer danger) exactly after START bit of response is detected by the eSDHC. While the clock is gated, the eSDHC samples an incorrect value from the command line thus sampling an incorrect command index in the response and eventually generating a command CRC and index error.

Impact: This issue occurs under a rare condition. The data transfer itself is not impacted.

Workaround: On detecting a command CRC error (IRQSTAT[CCE]) or command index error (IRQSTAT[CIE]), perform error recovery and re-issue the command without data. If Auto CMD12 is enabled for data transfer then Auto CMD12 won't be issued by hardware, so software needs to issue it after data transfer completion.

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

A-004373: Host may not detect SD card insertion/removal using DAT3 signal

Affects: eSDHC

Description: There is an on-chip pull-down resistor on the DAT3 signal that prevents the eSDHC host from correctly detecting card insertion or removal. The card has an internal pull-up of value 10-90K Ohms on the card detect pin, as defined in the SD specification. If the internal pull-up resistor value of the SD card is greater than 60KΩ, the card detection mechanism might not work properly.

Impact: Customer cannot use the sense DAT3 signal method for SD card detection.

Workaround: Do not use the DAT3 sensing method for card detection. Instead, use any of the following methods, as described in the application note for card detection provided by the SD Organization.

1. Using the CD pin on socket. This is a hardware solution. Note that this can be used only if the card socket provides the CD pin. Due to the on-chip pull-down on the I/O pin, a 10–50K Ohms pull-up resistor is needed for an active low polarity.
2. Polling SD memory card. This is a software solution. Check the response of the SEND_OP_CONDITION command to determine card insertion, and check the SEND_STATUS command to determine card removal.

Fix plan: No plans to fix

A-004577: PRSSTAT[DLA] bit does not reflect the data line state when any command with busy (R1b) is issued

Affects: eSDHC

Description: When an AutoCMD12 or any command with busy (R1b) is issued, PRSSTAT[DLA] bit should reflect the data line state. However, due to this erratum, PRSSTAT[DLA] is not applicable to detect data busy state. Furthermore, the corresponding transfer complete interrupt is not generated. However, the AutoCMD12 or any command with busy (R1b) can still be used with the restriction that busy needs to be de-asserted before sending new data command.

Impact: When an AutoCMD12 or any command with busy (R1b) is issued, PRSSTAT[DLA] bit does not reliably reflect the data line state.

Workaround: Software needs to wait for busy de-assertion before issuing any new data command. DAT0 line could be polled, but robust solution would be to keep sending CMD13(SEND_STATUS) until card reaches “trans" state.

- For AutoCMD12, CMD13 needs to be sent after TC of the data transfer command for which AutoCMD12 is enabled.
- For other command with busy, CMD13 needs to be sent after the command with busy completion(IRQSTAT[CC] = 1).

Fix plan: No plans to fix

A-005055: A glitch is generated on the card clock with software reset or a clock divider change

Affects: eSDHC

Description: A glitch may occur on the SDHC card clock when the software sets the SYSCTL[RSTA] bit (that is, performs a software reset). It can also be generated by setting the clock divider value. The glitch produced can cause the external card to switch to an unknown state. The occurrence is not deterministic and it happens rarely. The next command causes a timeout error(IRQSTAT[CTOE]) after this issue occurs.

Impact: Changing the frequency or performing a software reset for all may not work reliably.

Workaround: When the timeout error occurs for the command right after the SYSCTL[RSTA] is set or the clock divider value is changed, send CMD0 to bring the card to idle state, and perform re-initialization again. If the error occurs again, repeat this step until the initialization process completes.

Fix plan: No plans to fix

A-008358: Data timeout error may occur earlier then expected

Affects: eSDHC

Description: The data timeout counter value loaded into the timeout counter is less than expected and can result into early timeout error in case of eSDHC data transactions.

The table below shows the expected vs actual timeout period for different values of SYSCTL[DTOCV]:

Table 4. Expected vs actual timeout period for different values of SYSCTL[DTOCV]

DTOCV	Expected timeout	Actual timeout
0x0	2^13 SD_CLK periods	2^13 SD_CLK periods
0x1	2^14 SD_CLK periods	2^14 SD_CLK periods
0x2	2^15 SD_CLK periods	2^15 SD_CLK periods
0x3	2^16 SD_CLK periods	2^16 SD_CLK periods
0x4	2^17 SD_CLK periods	2^13 SD_CLK periods
0x5	2^18 SD_CLK periods	(2^17 – 2^16 + 2^13) SD_CLK periods
0x6	2^19 SD_CLK periods	(2^18 – 2^16 + 2^13) SD_CLK periods
0x7	2^20 SD_CLK periods	(2^19 – 2^16 + 2^13) SD_CLK periods
0x8	2^21 SD_CLK periods	2^13 SD_CLK periods
0x9	2^22 SD_CLK periods	(2^21 – 2^20 + 2^13) SD_CLK periods
0xA	2^23 SD_CLK periods	(2^22 – 2^20 + 2^13) SD_CLK periods
0xB	2^24 SD_CLK periods	(2^23 – 2^20 + 2^13) SD_CLK periods
0xC	2^25 SD_CLK periods	2^13 SD_CLK periods
0xD	2^26 SD_CLK periods	(2^25 – 2^24 + 2^13)SD_CLK periods
0xE	2^27 SD_CLK periods	(2^26 – 2^24 + 2^13) SD_CLK periods
0xF	Reserved	(2^27 – 2^24 + 2^13) SD_CLK periods

Impact: For data transactions, it may cause early timeout error in case of eSDHC data transactions.

Workaround: Use a higher value of SYSCTL[DTOCV] (even 0xF) to avoid early data timeout error.

Fix plan: No plans to fix

A-009204: System bus may hang if software register SYSCTL[RSTD] is set while DMA transactions are active

Affects: eSDHC

Description: In the event of that any data error (like, IRQSTAT[DCE]) occurs during an eSDHC data transaction where DMA is used for data transfer to/from the system memory, setting the SYSCTL[RSTD] register may cause a system hang. If software sets the register SYSCTL[RSTD] to 1 for error recovery while DMA transferring is not complete, eSDHC may hang the system bus. This happens because the software register SYSCTL[RSTD] resets the DMA engine without waiting for the completion of pending system transactions.

Impact: Causes system bus to hang.

Workaround: Add a 5 ms delay before setting SYSCTL[RSTD] to make sure all the DMA transfers are finished.

Fix plan: No plans to fix

A-007207: TBI link status bit may stay up after SGMII electrical idle is detected

Affects: Ethernet

Description: The TBI Status register (SR) contains a Link Status bit (TBI SR [Link Status]) that represents the current state of the SGMII link. If auto-negotiation (AN) is disabled, the TBI Link Status bit should be 1 (indicating the link is up) after recognizing IDLE sequences, and stay at 1 as long as valid data is received and the TBI is not reset. The TBI Link Status bit should be 0 (indicating the link is down) after several invalid characters are received or the TBI is reset. If AN is enabled, the TBI Link Status bit is not set to 1 until auto-negotiation is complete (TBI CR [AN DONE] = 1), but the same conditions as AN disabled then apply for the TBI Link Status bit to be cleared to 0.

An electrical idle (common mode) condition on the SGMII link results in the reception of invalid data and should cause the TBI Link Status bit to get cleared. If the transition from active to common mode takes enough time that the Rx is able to recognize at least 4 more K28.5 characters (for IDLE sequences, 70-80 UI), the portion of the design intended to detect the link down condition may shut off before the link down condition is actually reflected in the TBI.

This premature shutdown may cause the TBI Link Status to remain set to 1, indicating the link is up. This 'stuck at 1' condition persists until valid K28.5 characters are received again.

Impact: If the system never enters SGMII electrical idle, or if the transition from active to common mode takes less than 40 UI (~32 ns), then there is no impact and the false link up scenario does not occur.

If the system can generate an SGMII electrical idle condition as described above, then the TBI status may stay stuck at 1 while the link is down and does not transition to 0 until valid K28.5 characters are received again.

Workaround: If TBI SR[Link Status] = 0, the link is down.

For affected systems, there is no software access to the status of the electrical idle detection circuitry in SGMII mode and there is no bit replacement for TBI SR[Link Status] to monitor.

It may be more suitable to rely on other devices in the system for accurate link information by a polling or interrupt driven mechanism.

If it is required to monitor the link status from the affected device, the only feasible option is to monitor the progress of the Ethernet controller. When the TBI link status is set, the SW can periodically poll the state of the Ethernet controller by reading RBYT (receive byte counter). If the controller is expected to receive data packets, RBYT should increment. If RBYT has not incremented over a period of time it could indicate the link is down. However, there are various reason why RBYT may not increment (controller configuration errors, SerDes PLL issues, MIB Counter RCDE incremented, etc.). Examination of system conditions may be necessary to determine why RBYT has not incremented.

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:

- 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.

- 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] or halting the queues 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.

In case of platform errors, the controller will transmit corrupted system 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.

Transmit buffer descriptor work around:

For recovery from error scenarios 1:

If error interrupt handlers cannot resolve address or data errors without changing Tx state (e.g. BD address), execute a Tx reset to recover from Tx livelock condition.

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 error 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: Fixed in Rev 2.0

eTSEC 3: Multiple BD frame may cause hang

Description: In the “Transmit Data Buffer Descriptors (TxBD)” section of the device reference manual, it states the following:

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 the Destination Address (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 8: 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 9: Ethernet controller does not exit from Magic Packet mode when an oddly formed Magic Packet is received

Description: The Ethernet MAC should recognize as a Magic Packet any Ethernet frame with the following contents:

- A valid Ethernet header (Destination and Source Addresses)
- A valid FCS (CRC-32)
- A payload that includes the specific MagicPacket 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.

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

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

- 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.

- 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.

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

Workaround: There is no on-chip software or hardware workaround that can be performed to avoid or recover from this behavior. The controller does not wake up if one of these oddly formed magic packets is received, but any subsequent, properly formatted magic packet does wake up the controller.

If the received magic packet is properly formed, this erratum is avoided.

Fix plan: No plans to fix

eTSEC 10: 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 DA/SA (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)

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.

Impact: A packet with a non-matching Destination Address may be incorrectly recognized as a Magic Packet.

Workaround: None

Fix plan: Fixed in Rev 2.0

eTSEC 12: 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: Fixed in Rev 2.0

eTSEC 13: Odd prescale values not supported

Description: The IEEE Std 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

eTSEC 14: 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: Fixed in Rev 2.0

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: 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: Fixed in Rev. 2.0

eTSEC 17: 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.

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

eTSEC 19: Read of MIB counters may return zeros if ECNTRL[AUTOZ]=1

Description: If ECNTRL[AUTOZ]=1, then whenever software reads an MIB statistics counter,hardware will clear the counter results and starts incrementing again from 0. The hardware is supposed to return the previous contents of the statistics count er to the software. If the software read to an Rx MIB counter, however, occurs on the same cycle as the end-of-frame indicator for a Tx frame, or

the software read to a Tx MIB counter occurs on the same cycle as the end-of-frame indicator for an Rx frame, then the hardware will return zeroes instead.

Impact: Customers which use the AUTOZ feature of the MIB statistics counters may lose the data accumulated for a MIB counter since the last software read, if the end of a transmitting/ receiving frame coincides with the new software read..

Workaround: Disable auto-zeroing of MIB statistics counters by setting ECNTRL[AUTOZ]=0.

Fix plan: Fixed in Rev 2.0

eTSEC 20: 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: Fixed in Rev 2.0

eTSEC 22: Data corruption may occur in SGMII mode

Description: When operating the Ethernet controller in SGMII mode (ECNTRL[SGMIIM] = b'1), the device's SerDes's clock and data recovery may not accurately track the data if the incoming bit stream's data rate is slower than the expected data rate derived from the SD*n*_REF_CLK. This may cause data corruption in the received packet. The probability of failure is higher if SGMII is operating in 10-Mbps or 100-Mbps modes.

For example, if the SGMII SD*n*_REF_CLK is created by a 125 MHz ± 25 ppm oscillator, and the SGMII link partner reference clock is created by a separate 25 MHz ± 25 ppm oscillator, the incoming data rate may be slower, and this erratum applies.

Note that if the SGMII SD*n*_REF_CLK and the SGMII link partner reference clocks are created by a single clock oscillator, this erratum does not apply.

Impact: Customer systems that have an SGMII incoming bit stream data rate that is slower than the expected data rate derived from the SD*n*_REF_CLK may receive corrupted data in a packet that results in a CRC error that sets RxBD[CR] = 1 and increments the MIB counter RCDE.

When the packet is corrupted by this erratum and the CRC error is posted, one or more of the following may also occur:

- The packet may be truncated. This packet truncation may occur when an 8b10b symbol is incorrectly interpreted as a control character which forces end of packet.
- The SGMII link may go down until the next interpacket gap. While the link is down, eTSEC will transmit idles. If auto-negotiation is turned on (TBI MIIM Control Register [AN Enable] = 1), the eTSEC will attempt to auto-negotiate the SGMII link.
- The subsequent packet may be silently dropped.

After these events, normal data reception resumes.

Workaround: Use one of the following options:

- Use one clock oscillator to drive both the device SGMII SD*n*_REF_CLK and the link partner reference clock. There is an option to use a single ended clock for the SD*n*_REF_CLK. Refer to the device hardware specifications for details on single ended versus differential SD*n*_REF_CLK.
- Use a clock oscillator for the device's SGMII SD*n*_REF_CLK that has a frequency offset in the range of 10 to 200 ppm less than the SGMII link partner clock oscillator.

  For example: use a clock oscillator for SD*n*_REF_CLK which is specified at {125 MHz – 75 ppm} ± 25ppm = 124.990625 MHz ± 25 ppm, with a separate clock oscillator for the SGMII link partner, which is specified at 25 MHz ± 25 ppm. This combination creates a frequency offset of 25 to 125 ppm, with the incoming bit stream's data rate guaranteed to be faster than the expected data rate derived from the SD*n*_REF_CLK.

Fix plan: Fixed in Rev 2.0

eTSEC 23: 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: Fixed in Rev 2.0

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: Fixed in Rev 2.0

A-006293: Mixing TOE = 0 and TOE = 1 frames may cause data corruption

Affects: eTSEC

Description: eTSEC supports several TCP offload functions on transmitted frames, including IP checksum generation, TCP checksum generation, and VLAN tag insertion. If the controller is processing a mixture of frames with TxBD[TOE] = 1 and TxBD[TOE] = 0, it may use an incorrect Transmit Frame Control Block (TxFCB) for a frame and, therefore, incorrectly process the packet data.

The effects of incorrect TxFCB processing are as follows:

- TOE function executed on frame with TOE = 0
- TOE function not executed on frame with TOE = 1
- Different TOE function executed than intended (one or more fields of TxFCB perform an unintended action)

In most cases, this corrupts the frame in a way that is detected by the receiver (for example, checksum error). However, in some cases, the frame corruption may not be detected by the receiver.

Impact: Mixing TOE = 1 and TOE = 0 frames may cause corrupted packets.

Whether a particular frame is affected is dependent on the TxBD[TOE] settings and sizes of the frame(s) preceding the frame in question, as well as the transient state of the internal Tx FIFO SRAM.

Workaround: Do not mix TCP offload usage for frames. Write either TxBD[TOE] = 1 on all frames or TOE = 0 on all frames.

Fix plan: No plans to fix

A-006502: Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame

Affects: eTSEC

Description: Ethernet standards define the minimum frame size as 64 bytes. The eTSEC controller also supports receiving short frames less than 64 bytes, and can accept frames more than 16 bytes and less than 64 bytes 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 <= 2 bytes 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 fails 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 that is larger than 2 bytes resets the state so the graceful stop can complete (IEVENT[GRSC] = 1). A MAC Rx reset also resets the state.

In the second scenario, the parser and filer are enabled (RCTRL[PRSDEP] = 01,10,11). If a 1- or 1.5-byte frame is received, the controller carries 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 >= 2 bytes 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

A-006514: No parser error for packets containing invalid IPv6 routing header packet

Affects: eTSEC

Description: A packet with an IPv6 routing header with the following invalid conditions is not 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 does 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 is 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 is not flagged as parse error. L4 checksum result (such as ETU) may be invalid.

Workaround: Use one of the following options:

Option #1: Setup a MASK filer rule to allow only PID1.11 to participate in parser detection. For example, MASK = 0010_0000h. Next rule could file any matching packets to a user-specified physical or virtual queue. Subsequent rules can now follow planned rules.

Option #2: 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

A-007734: Stale time stamps and over-flow conditions can occur when using an external 1588 input clock at certain frequencies

Affects: IEEE 1588

Description: When the time stamp logic is enabled using an external 1588 input clock at certain frequencies, the eTSEC receiver can fail to update with the latest time stamp. This results in a stale time stamp in the Rx buffer padding even though eTSEC_TMR_RXTS is updated correctly.

In addition, the eTSEC receiver can hang because of over-flow conditions. The setting of bit[6] of eTSEC_RSTAT indicates the over-flow condition has occurred.

Impact: User cannot use the full frequency ranges of the 1588 input clock as specified in the data sheet:

- 17.85 MHz to 200 MHz for 1000 Mbps mode
- 3.57 MHz to 200 MHz for 100 Mbps mode
- 0.36 MHz to 200 MHz for 10 Mbps mode

Workaround: Limit the 1588 input clock frequency to:

- 66 MHz to 200 MHz for 1000 Mbps mode
- 13.2 MHz to 200 MHz for 100 Mbps mode
- 1.32 MHz to 200 MHz for 10 Mbps mode

Limit the maximum frequency of the 1588 input clock to the CCB frequency if it is lower than 200 MHz.

Fix plan: No plans to fix

GEN 1: eLBC PLL has incorrect default VCO selection

Description: The default setting for eLBC PLL VCO_SEL[1:0] is 0b'10 supporting a PLL output frequency from 133 MHz to 150 MHz. The default setting should be 0b'11 supporting a PLL output frequency from 83 MHz to 133 MHz. Note that the eLBC PLL is not enabled out of reset.

Impact: Low

Workaround: Software can update the LBIUPLLDCR0 Global Utilities register with the correct value of vco_sel when the PLL is enabled.

Fix plan: Fixed in Rev 2.0

GEN 2: Non-functional performance monitor events for ECM to DDR transactions.

Description: The following Performance Monitor events are non-functional.

eca_pm_30: ECM dispatch to DDR

eca_pm_84: ECM cancel due to DDR write TQ full

Workaround: None

Fix plan: No plans to fix

GEN-A004: READY_P1, READY_P0 Functionality

Description: In Rev 1.0 silicon the AUTORSTSR[READY_P1] and AUTORSTSR[READY_P0] fields and the external device output signals READY_P1 and READY_P0 could be asserted while the associated e500 core had not reached its ready state. This issue does not occur on device hard reset (HRESET assertion), but does occur when a core is reset via the MPIC PIR[P1] or PIR[P0] fields.

Impact: Software change required from rev. 1.0 to rev. 2.0 if:

- The cores are reset via the MPIC PIR[P1] or PIR[P0] fields and
- The system depends on AUTORSTSR[READY_P0], AUTORSTSR[READY_P1] or external READY_P0, READY_P1 pins to indicate that the associated core is in a ready.

Workaround: The following core1 (core0 is similar) reset procedure applies to both Rev 1.0 and 2.0 silicon.

1. Set PIR[P1] (to reset Core1)
2. Clear PIR[P1]

The above procedure also applies if one core is used to reset another one.

Fix plan: Fixed in Rev. 2.0

GEN-A007: Momentary glitch on I/Os during HRESET assertion and de-assertion

Description: The LVCMOS I/O pins may glitch momentarily during HRESET assertion and deassertion. The conditions for this glitch are as follows:

- Condition # 1: During HRESET assertion, all I/O pins listed in the table below may glitch periodically. The glitch time can be up to 1ns during a Sysclk clock cycle. The glitches can periodically occur every Sysclk cycle for a time period from 7us to 11us.
- Condition # 2: After HRESET de-assertion and before the READY signal is asserted, certain I/O pins as highlighted in the table below may glitch one time for a duration of up to 1ns.

Pin Name	Type	Condition # 1	Condition # 2
LAD[00:15]	IO	Y	Y
LDP[00:01]	IO	Y	Y
LA[16:31]	O	Y	
$\overline{\text{LCS}}$[00:04]	O	Y	
$\overline{\text{LCS}}$[05]/$\overline{\text{DMA2\_DREQ}}$[01]	O	Y	Y
$\overline{\text{LCS}}$[06]/$\overline{\text{DMA2\_DACK}}$[01]	O	Y	
$\overline{\text{LCS}}$[07]/$\overline{\text{DMA2\_DDONE}}$[01]	O	Y	
$\overline{\text{LWE}}$[00:01]	O	Y	
LBCTL	O	Y	
LALE	O	Y	
LGPL[00]/LFCLE	O	Y	
LGPL[01]/LFALE	O	Y	
LGPL[02]/$\overline{\text{LOE}}$/$\overline{\text{LFRE}}$	O	Y	
LGPL[3]/$\overline{\text{LFWP}}$	O	Y	
LGPL[04]/$\overline{\text{LGTA}}$/LFRB/LUPWAIT	O	Y	Y
LGPL[05]	O	Y	
LCLK[00:01]	O	Y	
LSYNC_IN	I	Y	Y
LSYNC_OUT	O	Y	
GPIO[00]/IRQ[07]	IO	Y	Y
GPIO[01]/IRQ[08]	IO	Y	Y
GPIO[02]/IRQ[09]	IO	Y	Y
GPIO[03]/IRQ[10]	IO	Y	Y
GPIO[04]/IRQ[11]	IO	Y	Y
GPIO[05]	IO	Y	Y
GPIO[06]	IO	Y	Y
GPIO[07]	IO	Y	Y
GPIO[08]/$\overline{\text{SDHC\_CD}}$	IO	Y	Y
GPIO[09]/SDHC_WP	IO	Y	Y
GPIO[10]/USB_PCTL0	IO	Y	Y

Table continues on the next page...

Pin Name	Type	Condition # 1	Condition # 2
GPIO[11]/USB_PCTL1	IO	Y	Y
GPIO[12:15]	IO	Y	Y
IIC1_SDA	IO	Y	Y
IIC1_SCL	IO	Y	Y
IIC2_SDA	IO	Y	Y
IIC2_SCL	IO	Y	Y
SDHC_DATA[00:03]	IO	Y	Y
SDHC_CMD	IO	Y	Y
SDHC_CLK	IO	Y	Y
SPI_MISO	I	Y	Y
SPI_MOSI	IO	Y	Y
SPI_CLK	IO	Y	
$\overline{\text{SPI\_CS}}$[00]/SDHC_DATA[04]	IO	Y	Y
$\overline{\text{SPI\_CS}}$[01]/SDHC_DATA[05]	IO	Y	Y
$\overline{\text{SPI\_CS}}$[02]/SDHC_DATA[06]	IO	Y	Y
$\overline{\text{SPI\_CS}}$[03]/SDHC_DATA[07]	IO	Y	Y
EC_MDC	O	Y	
EC_MDIO	IO	Y	Y
TSEC_1588_CLK_IN	I	Y	Y
TSEC_1588_TRIG_IN1	I	Y	Y
TSEC_1588_TRIG_IN2	I	Y	Y
TSEC_1588_ALARM_OUT01	O	Y	
TSEC_1588_ALARM_OUT02	O	Y	
TSEC_1588_CLK_OUT	O	Y	
TSEC_1588_PULSE_OUT[01]	O	Y	
TSEC_1588_PULSE_OUT[02]	O	Y	
TSEC1_TXD[07]/TSEC3_TXD[03]	O	Y	
TSEC1_TXD[06]/TSEC3_TXD[02]	O	Y	
TSEC1_TXD[05]/TSEC3_TXD[01]	O	Y	
TSEC1_TXD[04]/TSEC3_TXD[00]	O	Y	
TSEC1_TXD[00:03]	O	Y	
TSEC1_TX_EN	O	Y	
TSEC1_RX_ER	I	Y	Y
TSEC1_TX_CLK	I	Y	Y
TSEC1_GTX_CLK	O	Y	
TSEC1_CRS/TSEC3_RX_DV	IO	Y	Y
TSEC1_COL/TSEC3_RX_CLK	I	Y	Y
TSEC1_RXD[07]/TSEC3_RXD[03]	I	Y	Y
TSEC1_RXD[06]/TSEC3_RXD[02]	I	Y	Y
TSEC1_RXD[05]/TSEC3_RXD[01]	I	Y	Y
TSEC1_RXD[04]/TSEC3_RXD[00]	I	Y	Y

Table continues on the next page...

Pin Name	Type	Condition # 1	Condition # 2
TSEC1_RXD[00:03]	O	Y	Y
TSEC1_RX_DV	I	Y	Y
TSEC1_RX_ER	I	Y	Y
TSEC1_RX_CLK	I	Y	Y
EC_GTX_CLK125	I	Y	Y
TSEC2_TXD[07]	O	Y	
TSEC2_TXD[06]	O	Y	
TSEC2_TXD[05]/TSEC3_TX_EN	O	Y	
TSEC2_TXD[04]/TSEC3_GTX_CLK	O	Y	
TSEC2_TXD[00:03]	O	Y	
TSEC2_TX_EN	O	Y	
TSEC2_TX_ER	O	Y	
TSEC2_TX_CLK	I	Y	Y
TSEC2_GTX_CLK	O	Y	
TSEC2_CRS/TSE3_RX_ER		Y	Y
TSEC2_COL/TSEC3_TX_CLK	I	Y	Y
TSEC2_RXD[00:07]	I	Y	Y
TSEC2_RX_DV	I	Y	Y
TSEC2_RX_ER	I	Y	Y
TSEC2_RX_CLK	I	Y	Y
$\overline{\text{UART0\_RTS}}$	O	Y	
$\overline{\text{UART1\_RTS}}$	O	Y	
UART0_SOUT	O	Y	
UART1_SOUT	O	Y	
$\overline{\text{UART0\_CTS}}$	I	Y	Y
$\overline{\text{UART1\_CTS}}$	I	Y	Y
UART0_SIN	I	Y	Y
UART1_SIN	I	Y	Y
USB_D[00:07]	IO	Y	Y
USB_DIR	I	Y	Y
USB_NXT	I	Y	Y
USB_PWRFAULT	I	Y	Y
USB_STP	O	Y	
USB_CLK	I	Y	Y

Impact: The P2020 internal state will not be impacted by the glitches. However, depending on the use case of the I/O pins and the termination scheme, the glitches may cause unwanted behavior to the companion devices connected to the P2020.

Workaround: None

Fix plan: Fixed in Rev 2.1

I2C 1: Enabling I²C could cause I²C bus freeze when other I²C devices communicate

Description: When the I²C controller is enabled by software, if the signal SCL is high, the signal SDA is low, and the I²C 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 I²C 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 I²C bus to freeze. However, it happens very rarely due to the need for two conditions to occur at the same time.

Impact: Enabling the I²C controller may cause the I²C bus to freeze while other I²C devices communicate on the bus.

Workaround: Use one of the following workarounds:

- Enable the I²C controller before starting any I²C communications on the bus. This is the preferred solution.
- If the I²C 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 I²C 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 I²C registers if necessary.
  e. Go back to Step a.
  f. Normal operation.

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: Fixed in Rev 2.1

A-008312: Duplicate edge-triggered interrupt after priority rearbitration

Affects: MPIC

Description: There is an occurrence of duplicate interrupt when an edge-triggered interrupt higher in priority comes closely to any other enabled interrupts. The following is the sequence of events that leads to the duplicate edge-triggered interrupt:

1. An active interrupt is waiting for acknowledgement
2. An edge-triggered interrupt of higher priority triggers closely to the lower priority interrupt just when it is acknowledged
3. The higher priority edge-triggered interrupt supersedes and fires a new interrupt to the core
4. The core acknowledges the higher priority interrupt without clearing the pending state and finishes the interrupt service routine with EOI
5. A duplicate of the higher priority edge-triggered interrupt is triggered because of the uncleared pending state

Impact: Enabling any edge-triggered interrupts higher in priority than other enabled interrupts may lead to the duplicate edge-triggered interrupt. This includes edge-triggered IRQs, global timers and IPI.

Workaround: Choose one of the following workarounds based on the interrupt type:

- Configure the higher priority interrupts as level-sensitive only
  a. In case of IRQs this can be configured in the Vector/Priority Register.
  b. It is not an option for global timers or IPI.
- Any enabled edge-triggered interrupts must be no higher in priority than the other enabled interrupts.

Fix plan: No plans to fix

PCI-Ex 1: 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).

Impact: PCI Express controller may observe excess correctable errors when trained as a x2 link.

Workaround: None

Fix plan: Fixed in Rev 2.0

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: Fixed in Rev 2.0

PIC 1: NCPU of Feature reporting register does not reflect number of cores supported by device (P2010 only).

Description: The NCPU bit field of the Feature Reporting Register should report the number of cores supported by the device minus one. The current implementation is hard coded to 0b01 (two cores).

Impact: Number of cores supported by device cannot be obtained from PIC Feature Reporting Register.

Workaround: The software routine should look at the SVR to determine the product identity.

Fix plan: Fixed in Rev 2.0

PM 1: 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.

Fix plan: No plans to fix

SEC 1: STHA (Snow 3G Execution Unit) not reflected in the Controller EU Assignment Status Register (EUASR)

Description: When an EU is assigned to a channel by the controller, the EU assignment status register (EUASR) in the controller is updated to the EU that is assigned to the channel. In the case of the STHA (Snow3G Execution Unit), a new EU for the SEC 3.1, the assignment status register never shows that the STHA is assigned to a channel, even when it is.

Impact: The EU assignment status register is a legacy debug reference register. It only shows a snapshot in time of which EUs were owned by which channels. There is no identified use for EUASR outside debugging software or hardware issues.

Workaround: In rare debug cases, checking the EUASR to determine which channel owns an EU at a moment in time can be useful. However, there are other ways to determine this, such as software checking the State field in the crypto channel status register (CSR).

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

SRIO 1: CCSR[PW] does not reflect hardware width of port

Description: The control command and status register contains a Port Width field to indicate the hardware width of the port. This field is currently hard-coded to 2'b01 (4-lane port) at reset and is not writable.

The device supports SRIO ports that are inherently one lane in hardware or one lane by SerDes configuration. In both those cases, the port width setting should be 2'b00, but it is not.

Impact: A training sequence takes longer than expected, because it starts out training as x4 and then trains down to x1. It also has the unexpected register setting.

Workaround: Depend on training to train down to x1 and have the software ignore the incorrect register setting.

Fix plan: No plans to fix

SRIO 2: Serial RapidIO Packets with errors are not ignored by the controller while in input-retry-stopped state

Description: The RapidIO 1.2 specification states, in part 6, section 5.6.2.1 “Input Retry-Stopped Recovery Process” that the input side of a port which retries a packet must immediately enter the input-retry-stopped state and while in this state it must discard the rejected packet without reporting a packet error and ignore all subsequently received packets.

All packet errors received after the RapidIO controller enters input-retry-stopped state but before it sees a restart-from-retry control symbol should be ignored, but are not. Since the packets with errors are not ignored, the controller enters an input-error-stopped state.

Note that some RapidIO switches have been observed to transmit a packet with an out-of-order AckID after receiving a packet-retry. Before the switch sends restart from-retry, a transmission of this out-of-order AckID causes frequent “packet with unexpected AckID” errors in devices which have entered an input-retry-stopped state due to a loaded system.

A loaded system is defined as one that has its RapidIO input buffers filled with outstanding transactions. An example to this is a system in which 5 initiators across a switch make constant back-to-back NREAD requests into the receiving device without waiting for ACKs from the RapidIO controller. This will eventually fill the input buffers and cause the receiving device to enter input-retry-stopped state.

Impact: Packets with errors which should be ignored in input-retry-stopped state are not ignored and reported as errors. This causes the controller to enter an input-error-stopped state that needs hardware error recovery procedures, triggers error counting, and generates error interrupts.

Workaround: For systems using affected switches the preferred workaround is to disable error rate counting for packets with unexpected AckIDs. This leaves the hardware error recovery sequence in place for all errors, as well as error counting for the most common events associated with link degradation (CRC or invalid characters). The system will still enter the input-error-stopped state and the input-error-stopped recovery process will still need to be done, but no error counting will occur, and no error interrupt will be generated by the controller.

To disable error rate counting for unexpected AckIDs clear PnERECSR[UA] (Port n Error Rate Enable Command and Status Register - Unexpected AckID).

Fix plan: Fixed in Rev 2.0

SRIO 3: Message unit cannot generate messages with priority 0

Description: The priority of outbound messages is controlled by the DTFLOWLVL field of the outbound message destination attributes register (OMnDATR). If the DTFLOWLVL field is set to 0, however, the RMU generates a message of priority 1 instead of priority 0. The priority of outbound message transactions is set as follows:

OMnDATR[DTFLOWLVL]

00 SRIO transaction priority of 1

01 SRIO transaction priority of 1

10 SRIO transaction priority of 2

11 Reserved

Note that OMnDATR is set by a register write in direct mode, or by programming the Destination Attributes bits in the buffer descriptor in chaining mode.

Impact: Outbound messages have a higher priority than expected if the programmed flow level is 00.

Workaround: If the system requires all request packets to be of the same priority, use a flow level of 01 in all outbound ATMUs and destination attributes registers.

Fix plan: No plans to fix

SRIO-A002: SRIO reset command does not result in device reset

Description: The RapidIO (SRIO) Interconnect Specification specifies the following (e.g. in Part 6 section 3.5.5.1):

"The reset-device command causes the receiving device to go through its reset or power-up sequence. All state machines and the configuration registers reset to the original power on states. "

The affected Freescale devices implement this specification by asserting the HRESET_REQ output when a reset command is received over SRIO. Unfortunately, the HRESET_REQ output is not asserted long enough for the external reset circuit to assert the $\overline{\text{HRESET}}$ input. The net effect is that the SRIO reset command does not result in a device reset.

The failure is only applicable when the product is configured as an agent.

Impact: A SRIO master attempts to reset an SRIO agent by sending four consecutive maintenance reset control packets to the SRIO endpoint. The HRESET_REQ signal should be asserted long enough for an external device to detect the request and respond with $\overline{\text{HRESET}}$. On the affected Freescale devices the HRESET_REQ signal is asserted only for one SRIO clock period and can be missed by the external reset logic.

Workaround: The external SRIO device can write to the Global Utilities register RSTRSCR[SW_RR], which causes the HRESET_REQ output to assert, resulting in a device hard reset.

Fix plan: No plans to fix

SRIO-A004: SRIO controller may incorrectly transmit or block responses when PmCCSR[OPE] = 0

Description: The output port enable [OPE] field of the SRIO port m control command and status register (PmCCSR) is defined as follows:

Output port transmit enable:

- 0 - port is stopped and not enabled to issue any packets except to route or respond to I/O logical MAINTENANCE packets. Control symbols are not affected and are sent normally. This is the recommended state after device reset.
- 1 - port is enabled to issue any packets

When PmCCSR[OPE] = 0, the SRIO controller does not follow the expected behavior. Instead, the controller behaves as follows:

- Outbound request packets will not be transmitted (maintenance response packets should be sent).

- As long as no outbound request packets are pending (sent to SRIO, but not to the link), then all response packets (format type 8 and 13) will be transmitted (only responses for maintenance packets should be sent).

- Any pending outbound request packets may prevent response packets from being transmitted, too (responses for maintenance packets should be sent).

Examples of non-compliant scenarios:

1. An inbound IO read request packet (format type 2), packet 'B', arrives from a remote device. If there are no pending outbound request packets, the outbound response packet (format type 13), generated from packet 'B', will be transmitted. SRIO should only respond to inbound maintenance request packets (format type 8).
2. An outbound maintenance request packet (format type 8) will not be transmitted. SRIO should transmit maintenance response packets (format type 8).
3. An outbound read request packet (format type 2), packet 'C', is pending in SRIO. Afterwards, an inbound maintenance request packet (format type 8), packet 'D', arrives from a remote device. There is one pending outbound request packet (packet 'C'), which blocks later responses, so the outbound maintenance response packet (format type 8), generated from packet 'D', will not be transmitted until software sets OPE=1. SRIO should be able to respond to inbound maintenance request packets (format type 8) while OPE=0.

Impact: When PmCCSR[OPE] = 0

- Inbound non-maintenance request packets may be responded to.

- If the device configuration allows outbound requests to be pending in the controller, inbound maintenance request packets may not be responded to.

- Outbound maintenance request packets will not be transmitted.

Workaround: Set PmCCSR[OPE]=1 to allow transmission of any request or response packets.

To allow responding to inbound maintenance requests when PmCCSR[OPE] = 0, prevent outbound request packets from being sent to the SRIO controller by, for example, disabling any LAWs with SRIO target.

Fix plan: No plans to fix

USB 1: 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 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: Fixed in Rev 2.0

USB 3: 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 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 4: 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 5: 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 6: 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 7: 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 8: 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 10: 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-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: Write Global Utilities Memory offset 0xE0F68 to 0

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

A-003827: DATA PID error interrupt issued twice for the same high bandwidth ISO transfer

Affects: USB

Description: When receiving an Isochronous OUT transfer for a High Bandwidth endpoint (MULT > 0), if one of the DATA PIDs is corrupted, the controller issues two interrupts for that transaction error, one in the current microframe to signal the DATA PID error, and one fulfillment error in the next microframe.

On the beginning of a new microframe (upon receiving a SOF), the DMA checks and reports the status of the ISO transfer completed in the previous microframe. If the number of data packets correctly received for an ISO RX transfer is greater than 0 but less than MULT, the controller issues a fulfillment error by setting the transaction error bit. When a DATA PID error occurs, this error interrupt is triggered.

Both the bad PID and fulfillment error set the transaction error bit in the dTD. This is an ambiguous situation for the application that may then stop the endpoint from receving valid data in the next microframe (if there is another one from the Host).

Impact: This issue results in a correct ISO data transaction getting discarded.

There is no impact for the application software because data delivery in ISO transfers is not guaranteed. If there is a data delivery failure because of errors, no retries are attempted.

In ISO transfers, there is no dependency of the data between data packets and if a transaction is discarded due to a transaction error, the ISO stream can continue to the next dTD. This is transparent for the application, as the application is required to be capable of handling transaction errors in ISO streams.

The only impact of this issue is that the application discards not only the data transaction for which the DATA PID error occurred but also the next transaction.

Workaround: None

Fix plan: No plans to fix

A-003829: Host detects frame babble but does not halt the port or generate an interrupt

Affects: USB

Description: A high speed ISO Device, connected downstream to a high speed hub connected to the USB host, babbled in to the uframe boundary EOF1 time and the hub disabled the propagation of traffic to the upstream root host.

Inside the host controller, the ehci_ctrl state machine issues a request to the protocol engine to initiate the next transaction but this transaction is not sent to the USB as the port enable bit has been cleared. The result is that the ehci_crl state machine waits for the transaction to complete (which does not occur).

Eventually the software application times out without any frame babble error information. Just the iTD transaction error is issued.

The failure was seen with a high speed ISO device but a device babble error could occur on bulk or control or interrupt transactions for a failing device.

The final state is that the port has transitioned to full speed and the port enable bit is deasserted while the DMA does not know that there is a problem and the data structure shows only the occurrence of a transaction error.

This bug can occur for host IN transaction where the sending device under runs and error injects on the data packet. In this case the host may retry the IN immediately and it does not consider that the protocol engine may not be ready to issue the IN to the USB. The protocol engine issues the IN up to 5 useconds later than the EHCI Control state machine has issued the request to send the packet so it could result in a frame babble.

Impact: The host port is disabled, the halt bit is set, and the frame babble error is set in the associated data structure. This behavior complies with the EHCI specification for handling a frame babble error. The host controller driver handles the recovery by clearing the error conditions and re-queuing the transfer which should occur normally. When a frame babble occurs, there may be a loss of bandwidth because the application has to intervene and re-queue the transfer and re-enable the port.

Workaround: There is no workaround because the control of the retry is under hardware control so for non ISO transfers the hardware will retry so long as it determines that there is enough time but it does not account for the added delay due to the host protocol state machine being in the bus timeout state. Having a large TX FIFO and a good fill level (TXFIFOTHRES) will mean that there will be no under runs to host OUT transactions. This will significantly reduce the probability of occurrence of this issue for OUT Transactions. However please note that this bug can occur for host IN transaction where the sending device under runs and error injects on the data packet. In this case the host may retry the IN immediately.

Recovery:

The host will not get any response. The recovery from this condition will depend on the software, but host it will eventually time out and reset the device. This is not critical as this issue will only occur if the device downstream of a HS HUB is out of spec. and generates a frame babble.

Fix plan: No plans to fix

A-003832: Device NAKs OUT transaction if the host misses a handshake and retries

Affects: USB

Description: After the last transaction of an OUT transfer, the device core accepts the data and sends the handshake (ACK for control transfers, NYET for bulk transfers).

If the host does not receive the handshake (possibly because of corruption), it retries the OUT transaction. However, the device continuously NAKs the transaction retries because it has retired the dTD and is now unprimed.

The correct behavior from the device should be to follow the data sync protocol, the device should ACK the transaction and should not transfer the new data to memory as it has done this already.

Impact: This errata is a rare occurrence. The errata could lead to USB deadlock.

Workaround: There is no workaround that can be implemented on the device (USBDR) side. Only the host can resolve the situation by using a NAK counter. Please refer to NAK Count (NAKCnt) and NAK Count Reload (RL) fields in the Queue Head for more details about setting the NAK counter.

Fix plan: No plans to fix

A-003834: Host ACKs data2 PID sent by bulk endpoint when it should send BTO

Affects: USB

Description: In response to an IN token from the host controller, the device sends data with DATA2 PID, which is invalid for a bulk endpoint. The host controller should actually send bus time out (BTO). Instead, the host controller sends an ACK response. However, the received data is discarded because the PID does not match the expected PID toggle and this is handled as if a duplicate packet has been received.

This issue only occurs if the last received data from the previous packet was 0x54. Otherwise, the host controller behaves as expected and replies with BTO to the received DATA2 packet.

Impact: If the device behaves according to the USB standard, this issue will never occur.

Workaround: A workaround is not necessary. If the device behaves according to the USB standard, this issue will never occur.

Fix plan: No plans to fix

A-003836: Host does not retire ISO transfer if the first or second packet in MULT sequence is short.

Affects: USB

Description: Important Note: Behavior from USB Device has to be out of USB specification for this issue to occur.

When the Controller is acting as a Host executing High-Bandwidth HS ISO IN transfers, all data packets from the Device should have the same size as the set Maximum Packet Length. However, if a Device returns a short packet (smaller than mpl or even 0 bytes), the Host does not retire the transfer, and keeps sending token INs.

According to Appendix D-2 of the EHCI specification, the host should retire the iTD with a transaction error and not issue any more IN tokens for this uframe entry in the iTD.

Host does NOT keep on sending forever. It will retire the iTD the end of the microframe.

Example.

Let’s consider a iTD with total bytes of 3072 bytes, max. packet size of 1024 and MULT=3. The expected data movement on the bus for that microframe is: IN – DATA2 with 1024 bytes IN – DATA1 with 1024 bytes IN – DATA0 with 1024 bytes

But if, for example, the device responds to the second IN with DATA1 and a data payload such as 1023 or any other size which is less than 1024 the following will occur:

IN – DATA2 with 1024 bytes

IN – DATA1 with 1023 bytes

IN – DATA0 with 1024 bytes

(Host will retire the iTD after receiving DATA0)

The device returned 1023 bytes with DATA1 and the host issued another IN and the device returned Data 0 with 1024 bytes.

The host retired the iTD with Active bit clear, no error status and a total bytes field of 3071 bytes which accounts for the received data of 1024 + 1023 + 1024 packets.

So, basically the problem is that the host would issue IN tokens until MULT became zero as long as the device returned the correct PID order for the MULT setting and there were no other errors, it did not check that the device sent a max packet size for DATA2 and DATA1.

If device did not have 3072 (in this case 2048 because DATA1 is short), it should not have responded with DATA2 PID in response to first IN packet. Let's assume device had 2047 bytes of data and Endpoint is configured with MULT=3, correct behavior from device would have been:

IN – DATA1 with 1024 bytes

IN – DATA0 with 1023 bytes

Host will not send 3rd IN packet if the device responded correctly.

Impact: This issue does not result in any kind of lockup that requires software intervention. The iTD will be retried at the end of the microframe.

Workaround: None

Fix plan: No plans to fix

A-003837: When operating in test mode, the CSC bit does not get set to 1 to indicate a change on CCS

Affects: USB

Description: When in test mode (PORTSCx[PTC] != 0000), the Connect Status Change bit (PORSTCx[CSC]) does not get set to 1 to indicate a change in Current Connect Status (PORTSCx[CCS]).

Impact: This only affects the compliance test mode. There is no functional impact.

Workaround: Perform software polling for changes in the CCS bit (instead of using CSC) when test mode is enabled.

Fix plan: No plans to fix

A-003838: Device ACKs a DATA1 PID after SETUP token

Affects: USB

Description: If a DATA1 PID is received after the SETUP token instead of a SETUP/DATA0 sequence, the device controller sends an ACK packet.

According to section 8.5.3 of the USB specification, when corrupt data is received during the SETUP/DATA0 sequence, no handshake is returned. Receiving DATA1 instead of DATA0 should be considered as data corruption.

This may lead to a deadlock situation. Because the device acknowledges the data sent with the wrong PID, the host does not retry the transaction. The device software continues to wait for an interrupt to be asserted by the controller after the SETUP stage completes.

The device core discards the data sent by the erroneous host and reports the incorrect PID condition to the DMA. To successfully complete the SETUP stage, the USB interrupt is not issued. Because the device application does not prime the endpoint for the next transaction (data phase of status stage), it continuously sends NAK for incoming OUT/IN tokens.

This situation is unlikely to occur because the host that issues a DATA1 PID after a SETUP PID is a behavior that violates the USB specification.

Impact: This may lead to a deadlock situation. Because the device acknowledges the data sent with the wrong PID, the host does not retry the transaction. The device software continues to wait for an interrupt to be asserted by the controller after the SETUP stage completes.

Workaround: Not required, as the issue occurs when host behavior violates the USB specifications. However, the software can implement a "timeout" for setup transfers in device mode. If the setup transfer does not complete within the defined time, re-prime the endpoint.

Fix plan: No plans to fix

A-003840: The CERR is not decremented, and the xact err bit is not updated on a NYET handshake to a SETUP

Affects: USB

Description: According to the EHCI 1.0 specification, section 4.15.1.1, when a NYET is received after a SETUP/DATA0 sequence, the host controller must decrement the CErr field and set the XactErr bit (both these actions must be performed in the qTD). According to the USB specification, this NYET device response is invalid.

Due to this erratum, the host controller implementation does not decrement the CErr field or the XactErr bit for the NYET response, as described in the EHCI specification. Instead, the host controller accepts the NYET response and continues the Control transfer just as an ACK has been received and no PING protocol is initiated. The application is not aware that a NYET response has been received during a SETUP/DATA0 sequence. From a protocol point of view the host should cancel the transfer.

Accepting a NYET is not a USB spec violation itself since the device is not allowed to respond NYETby USB spec., it is a robustness issue, while the behavior of notdecrementing the CERR counter is in violation of defined in EHCI specsection 4.15.1.1. which states that when a NYET is received after aSETUP/DATA0 sequence, the host controller must decrement the CErr fieldand set the XactErr bit.

According to section 8.5.1.1, Figure 8-28 of USB Specification 2.0, Only ACK is allowed for the SETUP/DATA0 sequence. The NYET handshake from the Device is not expected (it will not be there if device follows the standard). ACK is the only acceptable answer to SETUP. For this to occur the device would be violating the spec. at this point. The controller should treat this situation as an error and instead it accepts the NYET response and continues the Control transfer just as an ACK has been received and no PING protocol is initiated. The application is not aware that a NYET response has been received during a SETUP/DATA0 sequence.

Impact: The impact of this depends on the erroneous device behavior. The host will assume that it has successfully completed the setup commands. If the device responds to the future transaction correctly, there will not be an issue. However, if device does not respond correctly, host may have to reset and configure the device again.

Workaround: A workaround is not needed because the issue will not occur if device is in spec. However, if the device replies with the behavior as mentioned above, there is no workaround.

Fix plan: No plans to fix

A-003842: Device controller may count below 3ms when detecting a Suspend state

Affects: USB

Description: Section 7.1.7.6 of the USB 2.0 specification defines the device's suspend detection as "constant Idle state on their upstream facing bus lines for more than 3.0 ms."

The current implementation counts exactly 3ms when the PHY clock is an ideal 30MHz or 60Mhz. Because the UTMI specification allows for some variation in the clock (that is, +/-500ppm), a fast clock results in a less than 3ms count. This can impact compliance testing (see Section 4.4.1, "Embedded High Speed Host Electrical Test Procedure specification") when it is automated with a 3ms count.

Impact: No functional impact.

Workaround: No workaround within the system. However, for compliance testing, coarse measurement of the 3ms in a scope is enough for most compliance houses.

Fix plan: No plans to fix

A-003843: Non-double word aligned buffer address sometimes causes host to hang on OUT retry

Affects: USB

Description: The USB host controller operating in streaming mode may under run while sending the data packet of an OUT transaction. This under run may occur if there are unexpected system delays in fetching the remaining packet data from memory. The host controller forces a bad CRC on the packet, the device detects the error and discards the packet. The host controller then retries a bulk, interrupt, or control transfer if an under run occurs according to the USB specification.

Due to this erratum, it was found that the host controller does not issue the retry of the failed bulk OUT. It does not issue any other transactions except SOF packets that have incorrect frame numbers.

The second failure mode occurs if the under run occurs on an ISO OUT transaction and the next ISO transaction is a zero byte packet. The host controller does not issue any transactions (including SOFs). The device detects a suspend condition, reverts to full speed, and waits for resume signaling.

A third failure mode occurs when the host under runs on an ISO OUT and the next ISO in the schedule is an ISO OUT with two max packets of 1024 bytes each.

The host controller should issue MDATA for the first OUT followed by DATA1 for the second. However, it drops the MDATA transaction, and issues the DATA1 transaction.

Impact: The system impact of this bug is the same regardless of the failure mode observed. The host controller hangs, the ehci_ctrl state machine waits for the protocol engine to send the completion status for the corrupted transaction, which never occurs. No indication is sent to the host controller driver, no register bits change and no interrupts occur. Eventually the requesting application times out.

Workaround:
- Use a double word offset even though the EHCI specification allows a non-double word offset to be used as a current offset for buffer pointer page 0 of the qTD. Otherwise, this issue may occur.
- Use non-streaming mode to eliminate under runs.

Fix plan: No plans to fix

A-003844: Host does not retire iTD but issues remaining transaction in next uframe

Affects: USB

Description: This issue occurs in systems that use high bandwidth high speed ISO transactions where the USB completes at least one of the transactions for the current uframe of the iTD on the USB successfully. However the transaction does not complete on the system bus until the next uframe because of low system bandwidth.

The failure occurs in a system where 3 high speed ISO IN transactions are scheduled for every uframe for a given endpoint. A buffer data mismatch error occurs as the device correctly sends data for the next uframe and the host writes that data in to the wrong buffer. A similar failure is seen for an ISO Out transaction where the host core sends data that should have been sent in the previous uframe.

This defect occurs on iTD's with multiple transactions scheduled per uframe

Impact: Causes data corruption for the next uFrame.

Workaround: No workaround is needed since it will recover following the next uFrame. Lower number of transactions per uFrame can prevent problem from occurring.

Fix plan: No plans to fix

A-003845: Frame scheduling robustness-Host may issue token too close to uframe boundary

Affects: USB

Description: When the USB host encounters an under-run while sending a Bulk OUT packet, it issues a CRC error according to the specification. However, the retry never occurs on the USB and the host appears to hang; it does not send any further transactions including SOF packets. The device ultimately detects a suspend condition and defaults to full speed mode. This can also happen for IN transactions where the device encountered an under-run and sent BAD CRC. The host will retry in this case without checking for the time left in the current uFrame. The response from the device will cause frame babble in this case.

Impact: The host appears to be hang as it does not send any further packets.

System Impact: The host port is disabled, the halt bit is set, and the frame babble error is set in the associated data structure. This behavior complies with the EHCI specification for handling a frame babble error. The host controller driver handles the recovery by clearing the error conditions and re-queuing the transfer which should occur normally. When a frame babble occurs, there may be a loss of bandwidth because the application has to intervene and re-queue the transfer and re-enable the port.

Workaround: For OUT transactions: If the host controller TX under-runs can be avoided then the problem will not occur for OUT transaction. Using a larger value for TXSCHOH can avoid this issue for OUT transactions.

For IN transactions: Insure the USB device side does not into an under-run condition. There is no workaround from the host side. The host controller driver (software) should handle the recovery by clearing the error conditions and re-queuing the transfer which should occur normally and re-enabling the port. The software driver can get the system restarted in this case. However, it cannot prevent the frame babble from occurring.

Fix plan: No plans to fix

A-003846: Host does not pre-fill TxFIFO correctly when data buffer address is not word aligned

Affects: USB

Description: The host core must pre-fill the TxFIFO with the number of bursts of data specified in the TXFIFOTHRESH field of the TXFILLTUNING register before issuing the OUT token.

A corner case exists where the host core does not wait until the requested burst of data has been fetched in to the TxFIFO before issuing the OUT token, leading to an under-run error on the OUT transaction. Because of the number of retries, this can cause a loop situation.

If the data buffer address is not word aligned, an initial single access is performed to align the address before the remaining bursts are issued. This initial alignment is accounted as one burst for TXFILLTUNING purposes, which makes the actual number of pre-fill bursts as TXFIFOTHRESH minus 1.

Impact: Because of retries done by USBDR, this issue can cause system to go in loop.

Workaround: Software should program data buffer addresses to be word aligned.

Fix plan: No plans to fix

A-003848: ISO IN - dTD not retired if MULT field is not correctly set

Affects: USB

Description: If the MULT field is wrongly set to a larger value than the expected number of packets from the dTD, the Controller will send zero length packets to all extra incoming IN tokens but will not retire the dTD. The active bit remains set. This erratum only applied to ISO type of USB device endpoints.

Impact: The dTD is not retired for ISO type of USB endpoints if MULT field not correctly set.

Workaround: As a software workaround, correct MULT should be used, matching the number of packets to be transmitted in a given dTD.

Fix plan: No plans to fix

A-003849: USB FORCE_ENABLE_LS feature not supported

Affects: USB

Description: The USBDR can be forced to work in low speed mode (in host mode only) using the PORTSCx[PTC] field. This feature is not working.

Impact: No impact. This feature is for testing purposes only.

Workaround: None

Fix plan: No plans to fix

A-003850: A write operation to PERIODICLISTBASE hangs system bus if PHY is in low power mode

Affects: USB

Description: When writing to the PERIODICLISTBASE register, if the PHY is in low power mode or USB PHY clock is not present, the system bus may hang because of a PHY clock dependency.

Impact: PERIODICLISTBASE register cannot be written if USB PHY clock is not present or during suspend mode.

Workaround: Do not write data to PERIODICLISTBASE if the PHY is set to low power mode. That is, do not write data to PERIODICLISTBASE if PORTSCx[PHCD]=1.

Fix plan: No plans to fix

A-004477: ULPI Function Control Register write gets corrupted by a soft reset

Affects: USB

Description: When a controller software reset (USBCMD[RST]) is issued at the data stage of a register write command to the ULPI Function Control Register (that is, after 0x84 is sent), the data bus changes to 8'h0 (reset value) and prevents the STP at the end of the command from being sent.

When the PHY does not receive the STP signal due to the reset, the ULPI Function Control Register may become corrupted. If a value of 0x0 is written to the ULPI Function Control Register, the SuspendM bit is set to 0, causing the PHY to enter the low-power mode which stops the clock while asserting DIR. The PHY may fail to accept the reset register write and remain in low-power mode.

Impact: Corruption of the ULPI Function Control Register may result in PHY clock being stopped. Writes to registers that require the PHY clock to run may hang.

Workaround: To prevent this erratum occuring, a reset should be performed only when the Run/Stop bit is disabled (USBCMD[RS] = 0):

- In Host Mode, software should not set the USBCMD[RST] bit to a 1 when USBSTS[HCH] = 0.
- In Device Mode, USBCMD[RST] should only be set to 1 when all primed endpoints have been flushed and the USBCMD[RS] bit is set to 0. This ensures that the device is not in an attached state before initiating a controller reset.

Fix plan: No plans to fix

A-005375: Full speed 'J' driven in host mode while doing remote wakeup

Affects: USB

Description: The host controller responds to a remote wakeup sequence by driving full speed 'J' for 1 full speed bit time. As the device is driving full speed 'K', there is a bus contention for 1 FS bit time. After this, the host controller drives 'K' as expected for the remaining sequence.

The consequences of this bus contention depend on each USB PHY implementation and on how the device USB PHY plus controller reacts to the contention.

This defect is seen only in UTMI+ serial interfaces in full speed.

This will not affect any device running UTMI in 8 or 16 bit mode.

Devices that have ULPI-only interfaces are not affected by this erratum.

Impact: There is a bus contention for 1 FS bit time when the host controller responds to a remote wakeup sequence by driving full speed 'J' for 1 full speed bit time.

Workaround: None

Fix plan: No plans to fix

A-005697: Suspend bit asserted before the port is in Suspend state

Affects: USB

Description: The EHCI specification states the following in the SUSP bit description:

In the Suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB.

In the USBDR controller, the PORTSCx[SUSP] bit changes immediately when the application sets it and not when the port is actually suspended.

Impact: Even though the behavior of the USBDR violates the EHCI specification statement, it does not cause any functional issue as, according to the EHCI specification, the application must wait for at least 10 milliseconds after a port indicates that it is suspended before initiating a port resume using the Force Port Resume bit.

Workaround: The software driver must follow the EHCI specification by waiting for at least 10 ms after setting the PORTSCx[SUSP] bit.

Fix plan: No plans to fix

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

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