MPC8313E Chip Errata

This document describes all known silicon errata for the MPC8313E PowerQUICC II Pro integrated host processor, silicon revision 1.0 and 2.x.

NOTE : Errata designated in this document as IEEE1588_n relate to the protocol defined in the IEEE Std 1588™.

The following table provides a revision history for this document.

Table 1. Document Revision History

Revision	Date	Significant Changes
7	06/2014	• Renamed USB38 to A-003817 • Renamed eTSEC-A002 to A-006502 • Added USB errata A-003837, A-003827, A-003829, and A-003845 • Added eTSEC erratum A-007207
6	09/2011	• Added USB-A005, USB-A007, eTSEC-A004 , eLBC-A002, SEC-A001, USB-A003, CPU-A022 • Updated General 17, USB32, USB-A002, USB-A001
5	05/2010	Removed • eTSEC57 because it is identical to eTSEC13 • IEEE 1588_13 Renamed • PCI1 to PCI15, DMA1 to DMA2, and IPIC1 to IPIC2 Added • CPU6, CPU-A002, eTSEC73, eTSEC74, eTSEC75, eTSEC76, eTSEC78, eTSEC79, eTSEC-A001, eTSEC-A002, IEEE_1588_8, IEEE_1588_12, IEEE_1588_20, IEEE_1588-A001, eLBC1, eLBC-A001, PCI15, PCI19, PCI20, USB31, USB32, USB33, USB34, USB35, USB36, USB37, USB38, USB-A001, and USB-A002 Modified • SEC6, SEC7, eTSEC9, eTSEC19, eTSEC39, eTSEC42, eTSEC61, eTSEC62, IEEE_1588_9, IEEE_1588_18, General12, General17
4	10/2008	• Renamed TSEC_TMR_PPn to TSEC_1588_PPn in IEEE_1588_16. • Removed Column for "Projected Solution" from Table 4. • Added Mask Nos for Revision 2.0 & 2.1 in Table 3. • Modified eTSEC41, eTSEC44, eTSCE68.

Table continues on the next page...

Table 1. Document Revision History (continued)

Revision	Date	Significant Changes
		• Added Jtag5, General15, General16, General17, DDR24, DDR25, eTSEC57, eTSEC71, eTSEC72, USB25, USB26, USB27, USB28, USB29, and USB30. • Renamed PCI25 to PCI24, SEC3 to SEC8, SEC2 to SEC7, SEC1 to SEC6, DDR1 to DDR15, DDR2 to DDR16 and DDR3 to DDR17. • Renamed Genral1 to General10. • Added eTSEC68, eTSEC69, eTSEC70, eLBC5. • Modified PCI22 work around to 'Plan to fix tPCIXKH in revision 2.1 silicon and tPCKHOX in 2.0 silicon'. • Modified eTSEC43 work around by clarifying that SICRH[28] and SICRH[29] enable delay on TSEC1_GTX_CLK and TSEC2_GTX_CLK, respectively. • Added Rev. 2.1 row in Table 4 and Table 3. • Added IEEE 1588_14, IEEE 1588_16, and IEEE 1588_19 (rev. 2.1 errata). • Changed projected solution for PCI25 to fixed. • Added USB25 and eTSEC66. • Modified eTSEC65 description and work around. • Removed JTAG and reset from General11. • Modified USB18 description to state, 'But when we detach the device by giving a power-on reset/hard reset...' • Added eTSEC67.
3	3/2008	• Modified eTSEC10 • Removed DDR20. DDR20 was accidentally published based on unsubstantiated data at that time. • Changed "Projected Solution" of eTSEC46, eTSEC47, eTSEC48, DDR3, USB21, eLBC3, • Added ", connect both eTSECs to 3.3 V," to work around for eTSEC10. • Moved R1 to left of memory block in DDR21. • Modified General11 Projected Solution. Will not be fixed. • Modified IPIC1 • Added DDR22, PCI25.
2	1/2008	• Added eTSEC60, eTSEC62, eTSEC63, eTSEC64, eTSEC65, IEEE 1588_17, IEEE 1588_18 • Added section to work around for PCI22 • Removed eTSEC57 (duplicate of eTSEC13) • Clarified IPIC1 • Updated eTSEC38, eTSEC59, IEEE 1588_15 • Updated eLBC2; changed disposition to plan to fix
1	10/2007	• Added SEC 13, DDR 21, SPI 5, eLBC 3, eTSEC 10, eTSEC 49, eTSEC 54, eTSEC 55, eTSEC 56, eTSEC 57, eTSEC 58, eTSEC 59, IEEE 1588_9, IEEE 1588_11, IEEE 1588_13. • Modified eTSEC 16 proposed solution to state that it will not be fixed in future revisions; parsing of tunneled IP packets will be disabled. • Modified eTSEC 40 title and work around to clarify that 333-MHz core frequency is required for gigabit operation. • Modified projected impact of IEEE 1588_3 to reduce confusion from 'exact point of a seconds rollover' to 'exact seconds rollover point'. • Clarified eTSEC 2 description, projected impact, and projected solution. Corrected eTSEC 2 description from ...speed modes 10/100/100 to ...speed modes 10/100/1000'. • Corrected TSEC1/2_GRX_CLK to TSEC1/2_GTX_CLK in eTSEC 43. • Removed eLBC 1. • Removed deprecated IEEE 1588_8 and replaced with IEEE 1588_15 (IEEE 1588_8 was a subset of IEEE 1588_15).
0	6/2007	Initial release.

The following table provides a cross-reference to match the revision code in the system version register to the revision level marked on the device.

Table 2. Revision Level to Part Marking Cross-Reference

Device	Silicon Mask			
	Rev 1.0	Rev 2.0	Rev 2.1	Rev 2.2
MPC8313E	M46H	M03R	M03R	M03R

The following table provides a values of system and processor version registers for different versions of silicons.

Table 3. Processor and System Version Register values

Device	Processor Version Register (PVR)	System Version Register (SVR)			
		Rev 1.0	Rev 2.0	Rev 2.1	Rev 2.2
MPC8313	0x8085_0010	0x80B1_0010	0x80B1_0020	0x80B1_0021	0x80B1_0022
MPC8313E	0x8085_0010	0x80B0_0010	0x80B0_0020	0x80B0_0021	0x80B0_0022

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 ‘No’ entry means it does not apply.

Table 4. Summary of Silicon Errata and Applicable Revision

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
CPU						
CPU6	DTLB LRU logic does not function correctly	No plans to fix	Yes	Yes	Yes	Yes
CPU-A002	CPU may hang after load from cache-inhibited, unguarded memory	No plans to fix	Yes	Yes	Yes	Yes
CPU-A022	CPU may hang after load from cache-inhibited, unguarded memory	No plans to fix	Yes	Yes	Yes	Yes
SEC						
SEC6	AES-CTR mode data size error	Fixed in Rev 2.0	Yes	No	No	No
SEC7	Single descriptor SRTP error	Fixed in Rev 2.0	Yes	No	No	No
SEC8	AES-CCM ICV checking write-back error	No plans to fix	Yes	Yes	Yes	Yes
SEC13	AESU/DEU initial reset requirement	No plans to fix	Yes	Yes	Yes	Yes
SEC-A001	Channel Hang with Zero Length Data	No plans to fix	Yes	Yes	Yes	Yes
DDR						
DDR15	DDR controller may wait two extra cycles between some commands	No plans to fix	Yes	Yes	Yes	Yes

Table continues on the next page...

Table 4. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
DDR16	DDR controller self refresh	No plans to fix	Yes	Yes	Yes	Yes
DDR17	DDR controller fails after initialization	No plans to fix	Yes	Yes	Yes	Yes
DDR19	DDR MDQ/MDM output setup with regards to MDQS (t_{DDKHDS}) does not meet the specification at 333 MHz	Fixed in Rev 2.0	Yes	No	No	No
DDR21	MCK/$\overline{\text{MCK}}$ AC differential crosspoint voltage outside JEDEC specifications	No plans to fix	Yes	Yes	Yes	Yes
DDR22	DRAMs using 12 x 8 x 2 configurations cannot be used in 16-bit bus mode	No plans to fix	Yes	Yes	Yes	Yes
DDR23	DDR read failure due to t_{DISKEW} spec violation	No plans to fix	Yes	Yes	Yes	Yes
DDR24	tDISKEW timing parameter may have insufficient margins	No plans to fix	Yes	Yes	Yes	Yes
DDR25	DDR controller does not meet the t_{DDKHAX} min value at 333 MHz	Fixed in Rev 2.1	Yes	Yes	No	No
eTSEC						
eTSEC1	eTSEC reads to filer do not work	Fixed in Rev 2.0	Yes	No	No	No
eTSEC2	eTSEC SGMII functionality is not available for any speed mode	Fixed in Rev 2.0	Yes	No	No	No
eTSEC3	eTSEC Parser does not properly parse L3 fields	Fixed in Rev 2.0	Yes	No	No	No
eTSEC5	WWR bit Anomaly	Fixed in Rev 2.0	Yes	No	No	No
eTSEC7	RMCA, RBCA counters do not correctly count valid VLAN tagged frames	Fixed in Rev 2.0	Yes	No	No	No
eTSEC8	RSTAT[RXF0] set regardless of destination ring if WWR=0	Fixed in Rev 2.0	Yes	No	No	No
eTSEC9	Error in arbitrary extraction offset	Fixed in Rev 2.0	Yes	No	No	No
eTSEC10	Limitations on the eTSEC1/eTSEC2 power supply connection	Fixed in Rev 2.0	Yes	No	No	No
eTSEC12	Tx IP and TCP/UDP Checksum Generation not supported for some Tx FCB offsets	Fixed in Rev 2.0	Yes	No	No	No
eTSEC13	Fetches with errors not flagged, may cause livelock or false halt	Partially fixed in Rev 2.0	Yes	No	No	No
eTSEC14	RMON results may be unreliable	Fixed in Rev 2.0	Yes	No	No	No
eTSEC15	Transmit jumbo frames greater than 2400 bytes may cause lost data, loss of BD synchronization, or false underrun error	Fixed in Rev 2.0	Yes	No	No	No
eTSEC16	Parsing of tunneled IP packets not supported	No plans to fix	Yes	Yes	Yes	Yes
eTSEC18	Parsing of MPLS label stack or non-IPv4/IPv6 label not supported	Fixed in Rev 2.0	Yes	No	No	No
eTSEC19	Compound filer rules do not roll back the mask	Fixed in Rev 2.0	Yes	No	No	No

Table continues on the next page...

Table 4. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
eTSEC20	Filer does not support matching against broadcast address flag PID1[EBC]	Fixed in Rev 2.0	Yes	No	No	No
eTSEC21	L3 fragment frame files on non-existent source/destination ports	Fixed in Rev 2.0	Yes	No	No	No
eTSEC22	RxBD[TR] not asserted during truncation when last 4 bytes match CRC	Fixed in Rev 2.0	Yes	No	No	No
eTSEC24	Parser results may be lost if TCP/UDP checksum checking is enabled	Fixed in Rev 2.0	Yes	No	No	No
eTSEC25	Transmission of truncated frames may cause hang or lost data	Fixed in Rev 2.0	Yes	No	No	No
eTSEC27	Transmitting PAUSE flow control frame may cause transmit lockup	Fixed in Rev 2.0	Yes	No	No	No
eTSEC28	eTSEC does not support parsing of LLC/SNAP/VLAN packets	Fixed in Rev 2.0	Yes	No	No	No
eTSEC29	Arbitrary extraction on short frames uses data from previous frame	Fixed in Rev 2.0	Yes	No	No	No
eTSEC30	Preamble-only with error causes false CR error on next frame	Fixed in Rev 2.0	Yes	No	No	No
eTSEC31	eTSEC filer reports incorrect Ether-types with certain MPLS frames	Fixed in Rev 2.0	Yes	No	No	No
eTSEC33	Parser does not check VER/TYPE of PPPoE packets	Fixed in Rev 2.0	Yes	No	No	No
eTSEC34	Some combinations of Tx packets may trigger a false Data Parity Error (DPE)	Fixed in Rev 2.0	Yes	No	No	No
eTSEC35	Back-to-back Rx frames may lose parser results of second frame	Fixed in Rev 2.0	Yes	No	No	No
eTSEC36	Generation of Ethernet pause frames may cause Tx lockup and false BD close	Fixed in Rev 2.0	Yes	No	No	No
eTSEC37	eTSEC half duplex receiver packet corruption	Fixed in Rev 2.0	Yes	No	No	No
eTSEC38	eTSEC Data Parity Error (DPE) does not abort transmit frames	Fixed in Rev 2.0	Yes	No	No	No
eTSEC39	May drop Rx packets in non-FIFO modes with lossless flow control enabled	Fixed in Rev 2.0	Yes	No	No	No
eTSEC40	Rx synchronization error may cause corrupted packets and limitations with Gigabit operation	Fixed in Rev 2.0	Yes	No	No	No
eTSEC41	Multiple BD Tx frame may cause hang	Fixed in Rev 2.0 if multiple Tx queues enabled.	Yes	No	No	No
eTSEC42	Frame is dropped with collision and HALFDUP[Excess Defer] = 0	No plans to fix	Yes	Yes	Yes	Yes
eTSEC43	eTSEC RGMII data to clock output skew at transmitter (t_{SKRGT}) does not meet the specification	Fixed in Rev 2.0	Yes	No	No	No

Table continues on the next page...

Table 4. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
eTSEC44	No parser error for packets containing invalid IPv6 routing header packet	Partially fixed in Rev 2.0	Yes	No	No	No
eTSEC45	eTSEC parser does not perform length integrity checks	Fixed in Rev 2.0	Yes	No	No	No
eTSEC46	eTSEC does not verify IPv6 routing header type field	Fixed in Rev 2.0	Yes	No	No	No
eTSEC47	No parse after back-to-back IPv6 routing header	Fixed in Rev 2.0	Yes	No	No	No
eTSEC48	L4 info passed to filer in L2/L3-only mode	Fixed in Rev 2.0	Yes	No	No	No
eTSEC49	MAC header data corruption	Fixed in Rev 2.0	Yes	No	No	No
eTSEC50	Rx_en synchronization may cause unrecoverable Rx errors at startup	Fixed in Rev 2.0	Yes	No	No	No
eTSEC54	Frames greater than 9600 bytes with TOE = 1 will hang controller	Fixed in Rev 2.0	Yes	No	No	No
eTSEC55	Arbitrary Extraction cannot extract last data bytes of frame	Fixed in Rev 2.0	Yes	No	No	No
eTSEC56	Setting RCTRL[LFC] = 0 may not immediately disable LFC	Fixed in Rev 2.0	Yes	No	No	No
eTSEC58	VLAN Insertion corrupts frame if user-defined Tx preamble enabled	No plans to fix	Yes	Yes	Yes	Yes
eTSEC59	False parity error at Tx startup	No plans to fix	Yes	Yes	Yes	Yes
eTSEC60	Tx data may be dropped at low system to Tx clock ratios	Fixed in Rev 2.0	Yes	No	No	No
eTSEC61	Rx may hang if RxFIFO overflows	Fixed in Rev 2.0	Yes	No	No	No
eTSEC62	Rx packet padding limitations at low clock ratios	Fixed in Rev 2.0	Yes	No	No	No
eTSEC63	False TCP/UDP checksum error for some values of pseudo header Source Address	Fixed in Rev 2.0	Yes	No	No	No
eTSEC64	User-defined Tx preamble incompatible with Tx Checksum	No plans to fix	Yes	Yes	Yes	Yes
eTSEC65	Transmit fails to utilize 100% of line bandwidth	Partially fixed in Rev 2.0	Yes	No	No	No
eTSEC66	Controller stops transmitting pause control frames	Fixed in Rev 2.1	No	Yes	No	No
eTSEC67	ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software	No plans to fix	Yes	Yes	Yes	Yes
eTSEC68	Half-duplex collision on FCS of Short Frame may cause Tx lockup	Documentation update	Yes	Yes	Yes	Yes
eTSEC69	Magic Packet Sequence Embedded in Partial Sequence Not Recognized	No plans to fix	Yes	Yes	Yes	Yes
eTSEC70	MAC: Malformed Magic Packet Triggers Magic Packet Exit	No plans to fix	Yes	Yes	Yes	Yes
eTSEC71	The value of TSEC_ID2 is incorrect	No plans to fix	No	Yes	Yes	Yes

Table continues on the next page...

Table 4. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
eTSEC72	Receive pause frame with PTV = 0 does not resume transmission	No plans to fix	Yes	Yes	Yes	Yes
eTSEC73	TxBD polling loop latency is 1024 bit-times instead of 512	No plans to fix	Yes	Yes	Yes	Yes
eTSEC74	MAC: Rx frames of length MAXFRM or MAXFRM-1 are marked as truncated	No plans to fix	No	Yes	Yes	Yes
eTSEC75	Misfiled Packets Due to Incorrect Rx Filer Set Mask Rollback	No plans to fix	No	Yes	Yes	Yes
eTSEC76	Excess delays when transmitting TOE=1 large frames	No plans to fix	Yes	Yes	Yes	Yes
eTSEC78	Controller may not be able to transmit pause frame during pause state	No plans to fix	Yes	Yes	Yes	Yes
eTSEC79	Data corruption may occur in SGMII mode	No plans to fix	Yes	Yes	Yes	Yes
eTSEC-A001	MAC: Pause time may be shorter than specified if transmit in progress	No plans to fix	Yes	Yes	Yes	Yes
A-006502	GRS may fail to complete after receiving a 1- or 2-byte frame	No plans to fix	Yes	Yes	Yes	Yes
eTSEC-A004	User-defined preamble not supported at low clock ratios	Fixed in Rev. 2.0	Yes	No	No	No
A-007207	TBI link status bit may stay up after SGMII electrical idle is detected	No plans to fix	Yes	Yes	Yes	Yes
IEEE1588						
IEEE 1588_1	eTSEC IEEE 1588 reset bit	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_2	IEEE 1588 not supported in SGMII mode	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_3	FIPER outputs cannot be phase aligned to a specific point in time	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_4	ALARM output does not go active unless the ALARM comparison time is exactly equal to the current time value	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_5	1588 external trigger timestamp does not include offset register value	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_6	1588 ALARM output comparison value does not include offset register value	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_7	ALARM output signal deasserts prematurely during current system time rollover condition	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_8	Writing Offset registers during use may yield unpredictable results	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_9	1588 reference clock limited to 1/2 controller core clock in asynchronous mode	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_10	eTSEC IEEE Std 1588 logic only requires one offset register per counter instance	Fixed in Rev 2.0	Yes	No	No	No

Table continues on the next page...

Table 4. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
IEEE 1588_11	1588 reference clock pulse required between writes to TMR_ALARMn_L and TMR_ALARMn_H	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_12	1588 alarm fires when programmed to less than current time	Partially fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_14	TxPAL timestamp uses TxBD snoop enable instead of Tx data	No plans to fix	No	Yes	Yes	Yes
IEEE 1588_15	Use of asynchronous 1588 reference clock may cause errors	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_16	Odd prescale values not supported	No plans to fix	No	Yes	Yes	Yes
IEEE 1588_17	Cannot use inverted 1588 reference clock when selecting eTSEC system clock as source	No plans to fix	Yes	Yes	Yes	Yes
IEEE 1588_18	FIPER's periodic pulse phase not realigned when the 1588 current time is adjusted	Fixed in Rev 2.0	Yes	No	No	No
IEEE 1588_19	Tx FIFO data parity error (DPE) may corrupt Tx timestamps if TMR_CTRL[TRTPE]=1	No plans to fix	No	Yes	Yes	Yes
IEEE 1588_20	eTSEC 1588: Write to reserved 1588 register space causes system hang	No plans to fix	No	Yes	Yes	Yes
IEEE1588-A001	Missing or incorrect received frame timestamp values occur when 1588 time-stamping is enabled	Plan to fix in Rev 2.2	Yes	Yes	Yes	No
eLBC						
eLBC1	LTESR[CS] error issue	Fixed in Rev 2.0	Yes	No	No	No
eLBC2	UPM does not have indication of completion of a Run Pattern special operation	Fixed in Rev 2.0	Yes	No	No	No
eLBC3	eLBC NAND Flash memory has an ECC syndrome that collides with the JFFS2 marker in Linux	Fixed in Rev 2.0	Yes	No	No	No
eLBC5	LTEATR and LTEAR may show incorrect values under certain scenarios	No plans to fix	Yes	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	Yes
eLBC-A002	Core may hang while booting from NAND using FCM	No plans to fix	Yes	Yes	Yes	Yes
PCI						
PCI2	NXT_PTR field of hot swap register in PCI power management	Fixed in Rev 2.0	Yes	No	No	No
PCI15	Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang	No plans to fix	Yes	Yes	Yes	Yes

Table continues on the next page...

Table 4. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
PCI19	Dual-address cycle inbound write accesses can cause data corruption	No plans to fix	Yes	Yes	Yes	Yes
PCI20	PCI controller may hang when returning from "PCI pins low" state	No plans to fix	Yes	Yes	Yes	Yes
PCI22	PCI input and output hold from clock (t_{PCIXKH} and t_{PCKHOX}) do not meet specification	Fixed in Rev 2.1	Yes	Yes	No	No
PCI23	Device locks up if a wake-up event occurs immediately after D3-warm instruction by e300	Fixed in Rev 2.0	Yes	No	No	No
PCI24	While ICACHE on, critical data word access is corrupted	Fixed in Rev 2.0	Yes	No	No	No
DMA						
DMA2	Data corruption by DMA when destination address hold (DAHE) bit is used	No plans to fix	Yes	Yes	Yes	Yes
IPIC						
IPIC2	eTSEC interrupts are swapped in the IPIC	Fixed in Rev 2.0	Yes	No	No	No
USB						
USB1	USB jitter issue	Fixed in Rev 2.0	Yes	No	No	No
USB15	Read of PERIODICLISTBASE after successive writes may return a wrong value in host mode	No plans to fix	Yes	Yes	Yes	Yes
USB18	USB shows device attached even when the device has been detached in full-speed mode	Fixed in Rev 2.0	Yes	No	No	No
USB19	USBDR in Host mode does not generate an interrupt upon detection of a CRC16/PID/Timeout error when a received IN data packet is corrupted	No plans to fix	Yes	Yes	Yes	Yes
USB20	Problem with configuration of RCWH/ CFG_RESET_SOURCE when ULPI intended	No plans to fix	Yes	Yes	Yes	Yes
USB21	SE0_NAK issue	Fixed in Rev 2.0	Yes	No	No	No
USB25	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	Yes
USB26	NackCnt field is not decremented when received NYET during FS/LS Bulk/ Interrupt mode	No plans to fix	Yes	Yes	Yes	Yes
USB27	When an ACK or NAK is sent from the device in response to a PING, the CERR counter value is not being reset to the initial value	No plans to fix	Yes	Yes	Yes	Yes
USB28	In device mode, when receiving a Token OUT, if a Rx flush command is issued at	No plans to fix	Yes	Yes	Yes	Yes

Table continues on the next page...

Table 4. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
	the same time, to the same endpoint, the packet will be lost					
USB29	Priming ISO over SOF will cause transmitting bad packet with correct CRC	No plans to fix	Yes	Yes	Yes	Yes
USB30	High speed output impedance fails marginally.	No plans to fix	Yes	Yes	Yes	Yes
USB31	Transmit data loss based on bus latency	No plans to fix	Yes	Yes	Yes	Yes
USB32	Missing SOFs and false babble error due to Rx FIFO overflow	No plans to fix	Yes	Yes	Yes	Yes
USB33	No error interrupt and no status will be generated due to ISO mult3 fulfillment error	No plans to fix	Yes	Yes	Yes	Yes
USB34	NAK counter decremented after receiving a NYET from device	No plans to fix	Yes	Yes	Yes	Yes
USB35	Core device fails when it receives two OUT transactions in a short time	No plans to fix	Yes	Yes	Yes	Yes
USB36	CRC not inverted when host under-runs on OUT transactions	No plans to fix	Yes	Yes	Yes	Yes
USB37	OTG Controller as Host does not support Data-line Pulsing Session Request Protocol	No plans to fix	Yes	Yes	Yes	Yes
A-003817	USB Controller locks after Test mode "Test_K" is completed	No plans to fix	Yes	Yes	Yes	Yes
USB-A001	Last read of the current dTD done after USB interrupt	No plans to fix	Yes	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	Yes
USB-A003	Illegal NOPID TX CMD issued by USB controller with ULPI interface	No plans to fix	Yes	Yes	Yes	Yes
USB-A005	ULPI Viewport not Working for Read or Write Commands With Extended Address	No plans to fix	Yes	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	Yes
A-003827	DATA PID error interrupt issued twice for the same high bandwidth ISO transfer	No plans to fix	Yes	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	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	Yes
A-003845	Frame scheduling robustness-Host may issue token too close to uframe boundary	No plans to fix	Yes	Yes	Yes	Yes
SPI						

Table continues on the next page...

Table 4. Summary of Silicon Errata and Applicable Revision (continued)

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	2.0	2.1	2.2
SPI5	Selection of GPIO functionality on $\overline{\text{SPISEL}}$ signal causes MME in SPI	Fixed in Rev 2.0	Yes	No	No	No
General						
General10	Incorrect isolation for mux control for LCLK0/LCLK1 in low power mode	Fixed in Rev 2.0	Yes	No	No	No
General11	DUART input high voltage does not meet the specification	No plans to fix	Yes	Yes	Yes	Yes
General12	CSB deadlock	Fixed in Rev 2.0	Yes	No	No	No
General15	Electrostatic discharge (ESD) may fail to meet the 500 V charged device model (CDM)	No plans to fix	No	No	Yes	Yes
I2C						
General16	Enabling I^2C could cause I^2C bus freeze when other I^2C devices communicate	No plans to fix	Yes	Yes	Yes	Yes
General						
General17	DUART: Break detection triggered multiple times for a single break assertion	No plans to fix	Yes	Yes	Yes	Yes
JTAG						
JTAG5	The JTAG fails to capture the correct values of the receive pins of SerDes Interface	No plans to fix	Yes	Yes	Yes	Yes

CPU6: DTLB LRU logic does not function correctly

Description: The DTLB is implemented as 2-way set associative with 32 entries per way. EA[15:19] is used to determine which one of the 32 entries of both ways. When a DTLB miss occurs, normally the CPU provides information (through SRR1 bit 14, DTLB replacement way) to indicate which of the two ways the software (DTLB exception handler or the software table walk routine) should use to bring in the new page. Presumably, the CPU provides the information, through SRR1 bit 14, based on the LRU (least recently used) algorithm. However, because of this bug, this is not the case.

In fact, for any given EA[15:19], when a DTLB miss occurs, SRR1 bit 14 always indicates that way1 should be used or replaced. (Except if it is the very first DTLB miss after hard reset. In this case, SRR1 bit 14 indicates way0 should be used.)

In other words, if the DTLB exception routine follows the SRR1 bit 14's suggestion to do the TLB replacement, it always replaces the one in way1. In addition, whatever has been loaded in way0 is effectively locked and is not replaced.

Impact: Performance degradation due to the reduction of usable DTLB entries.

Workaround: In the software, use one word (32 bits) to keep the record of the DTLB way that is Least Recently Written (LRW). Use this information to overwrite the SRR1 bit 14 (DTLB replacement way) when a Data Translation Miss exception occurs. Basically, the LRU (least recently used) hardware algorithm is changed to an LRW (least recently written) software algorithm.

For the system that has no secondary storage (such as a hard drive), it is highly recommended to set the C (Change) bit during the DTLB load exception to optimize the performance.

For a system that has a secondary storage and the OS does a page swap, the OS can choose whether to set the C bit in the DTLB load exception.

If the C bit is set in the DTLB load exception, it can preempt the subsequent DTLB store exception to the same page. However, since all the pages are marked as changed, during the page swap all pages must be written back to the secondary storage regardless of whether they have really been changed or not.

If the C bit is not set in the DTLB load exception with the LRW algorithm, a subsequent store to the same page as the previous load will use a separate entry. Therefore, one page occupies both ways. This causes inefficiency for the DTLB allocation but may save time during the page swap since the page change status is correctly marked.

Fix plan: No plans to fix

CPU-A002: CPU may hang after load from cache-inhibited, unguarded memory

Description: There is a one-core-clock-cycle window of opportunity for a snoop to collide with the data returned from cache-inhibited memory to a load that has been cancelled. This collision can hang the data cache in a busy state, prohibiting further data cache accesses. The snoop address is immaterial.

The load can be cancelled if it meets the following conditions: it was executing speculatively beyond a conditional branch that resolved unfavorably or was pre-empted by an external interrupt or decrementer interrupt that took priority. The load must be from a cache-inhibited, non-guarded memory block. A load from cacheable memory, even if it misses in the cache, does not hang the data cache in a busy state, and if the memory block is marked guarded, the load will not be executed out of order.

An external master must put addresses on the bus with the attribute of memory coherency required (Global) so that the CPU allows snooping of the data cache. External masters have to be programmed to snoop.

Impact: CPU halts or stops executing on a subsequent load or store that finds the data cache busy. This load or store does not complete, and the data cache stays busy until a hardware or software reset ($\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$). Debug tools will reveal that the program counter is stopped and pointing to the load or store that cannot complete.

Workaround: Use one of the following options:

- Set bit 14 in the HID2 register. This bit disables a minor feature that attempted to take advantage of cache hits under a cancelled cache miss which was still waiting for data. There should be no performance loss from disabling this feature. HID2[14] is not implemented on any other e300 devices. Setting it will have no effect on other devices or any future revisions.
- Mark all data memory blocks from which loads could occur “cacheable” or “guarded.”

Fix plan: No plans to fix

CPU-A022: The e300 core may hang while using critical interrupt

Description: If BOTH critical interrupt AND normal interrupt types are used in a system, the e300 core may hang.

Impact: The processor may stop dispatching instructions until a hardware reset($\overline{\text{HRESET}}$). Debug tools will not be able to read any register correctly except program counter IAR which points to a location in the critical interrupt vector.

Workaround: If both critical interrupt and normal interrupt types are used, then instead of using an **rfi** instruction at the end of every exception handler, replace the **rfi** with the following:

1. Disable critical interrupts by setting MSR[CE] to 0 with a **mtspr** instruction.
2. Copy SRR0 and SRR1 to CSRR0 and CSRR1, respectively.
3. Execute an **rfci** instruction. This enables MSR[CE] and any other bits that the original **rfi** would have set including the MSR[EE].

Sample Code:

```
// Disable MSR[CE]
mfmsr r2
lis r3, 0xffff
ori r3, r3, 0xff7f
and r2, r2, r3
sync
mtmsr r2
isync
// Copy SRR0, SRR1 to CSRR0 and CSRR1
mfspr r2, srr0
mfspr r3, srr1
mtspr csrr0, r2
mtspr csrr1, r3
...restore GPRs
rfci
```

Fix plan: No plans to fix

SEC6: AES-CTR mode data size error

Description: The SEC 2.1 supports acceleration of AES Counter mode, an underlying algorithm in Secure Realtime Transport Protocol (SRTP), and optionally, IPSec. The SEC is designed to accelerate AES-CTR alone (using descriptor type 0001_0) or in parallel with an HMAC-SHA-1 using a special SRTP descriptor type 0010_1. SRTP uses AES-CTR with HMAC-SHA-1. Although AES in counter mode (AES-CTR) is meant to act as a stream cipher, the AESU considers any input data size that is not an even multiple of 16 bytes to be an error.

Impact: None

Workaround: Use one of the following options:

- The AESU Data Size error can be disabled via the AESU Interrupt Control Register to prevent a nuisance interrupt.
- The input data length (in the descriptor) can be rounded up to the nearest 16B. Set the data-in length (in the descriptor) to include X bytes of data beyond the payload. Set the data-out length to only output the relevant payload (don't need to output the padding). SEC reads from memory are not destructive, so the extra bytes included in the AES-CTR operation can be whatever bytes are contiguously trailing the payload.

Fix plan: Fixed in Rev 2.0

SEC7: Single descriptor SRTP error

Description: The SRTP protocol specifies the use of AES-CTR with HMAC-SHA-1. The SEC 2.1 is designed to accelerate AES-CTR in parallel with HMAC-SHA-1 using a special SRTP descriptor type 0010_1. Single descriptor SRTP does not work for data sizes that are not even multiples of 16 bytes.

Impact: When operating on input data which is N*16B, the AESU and MDEU can each read in the portion of the datastream relevant to their respective operations. When the input data is not N*16B, the AESU data size workaround described in SEC5 (rounding up to the next 16B) does not work because these excess bytes are ‘snooped’ by the MDEU and corrupt the HMAC-SHA-1 operation.

Workaround: Because the SRTP protocol does not require the payload to be N*16B, most packets will likely be of a data length that hits this erratum. The workaround is to use two descriptors to perform SRTP.

Outbound: The first descriptor is type 0001_0 "common non-snoop" set for an AES-CTR operation using either of the workarounds described in SEC5. The second descriptor is type 0001_0 “common non-snoop” set for an HMAC-SHA-1 of the headers + unpadded encrypted payload + MKI (when present).

Inbound: Same descriptors as outbound, but in reverse order.

To minimize the performance impact of this workaround, these two descriptors should be created simultaneously and launched back-to-back. By configuring the SEC Crypto-channel to perform Done notification on selected descriptors, the first descriptor should be set to not generate a Done interrupt, while the second descriptor (which completes the SRTP operation) should be set to generate the Done interrupt. All the parameters required to build both descriptors are available at the start of the request to the SEC device driver, so there is no reason to wait for the first descriptor to complete before building and launching the second.

Fix plan: Fixed in Rev 2.0

SEC8: AES-CCM ICV checking write-back error

Description: Most security protocols add an authentication tag to the frame or packet which provides the receiver with proof that the frame/packet has not been modified in transit. These authentication tags are also known as an HMACs or ICVs. The SEC is capable of generating these authentication tags for out-bound operations, and re-calculating the tag and comparing it to the received tag for in-bound operations. When performing in-bound authentication tag comparison, the SEC can notify software of a miscompare (which indicates an intentional modification of a frame/packet) by generating a crypto-channel error interrupt, or by writing a tag miscompare bit to the original descriptor header in memory. For authentication tags (ICVs) created by the AESU when in CCM mode, hardware can successfully perform the ICV comparison, but signaling of an ICV miscompare can only be performed via a channel error interrupt. When attempting to use the writeback method when an ICV miscompare is detected, all subsequent descriptors will be flagged as having ICV miscompares, whether they do or not.

Impact: A full SEC reset is required to clear the ICV miscompare writeback logic.

Workaround: For AES-CCM mode only, do not use the header writeback method for signaling authentication tag miscompare. Use the channel interrupt method. Alternately, perform ICV checking in software, as is required in all SEC cores 2.0 and below.

Fix plan: No plans to fix

SEC13: AESU/DEU initial reset requirement

Description: The SEC 2.2 is a size optimized version of the Freescale SEC core. One size optimization is the AESU and DEU share a FIFO and other bus interface logic used in descriptor-based operations. There is a bug in the bus interface logic that causes the bus interface to lock out either the AESU or DEU if those blocks are not reset before descriptor based operations begin.

Impact: When the PowerQUICC device comes out of reset, the user needs to initialize the SEC, including configuration of registers in the controller, channel, and EUs. EU initialization operations primarily consist of masking off low level interrupts. This host access to the EUs causes the SEC to lock out either the DEU or AESU, so when the user switches to descriptor based operations, one of the EUs does not work.

This errata does not affect the run-time performance of the SEC 2.2 or reduce functionality. Avoiding the errata requires an extra step at initialization.

Workaround: Following host configuration of each EU, the host needs to perform a soft reset of each EU by writing the value “0x2” to the DEU/AESU Reset Control Register. This soft reset will reset most registers within the DEU/AESU, but will not clear the settings of the EU Interrupt Control Registers. This soft reset has the effect of resetting the bus interface to allow both the DEU and AESU to function properly in descriptor based operations.

Any further host access to the EU registers once descriptor based operations have begun requires an EU soft reset before continuing with descriptor based operations. Note that direct EU access once the EUs are configured is rarely needed. If an EU error causes the host to read EU registers for diagnosis and interrupt clearing, the host should include an EU soft reset as the final step in the interrupt service routine.

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

DDR15: DDR controller may wait two extra cycles between some commands

Description: When using 8-beat bursts (only supported for a 32-bit bus with DDR1 memory), the DDR controller may place 2 extra cycles between the following transactions:

- Read->Read to different chip select
- Read->Write for data bus contention
- Write->Read for data bus contention
- Write->Read to same chip select (to satisfy TIMING_CFG_1[WR_TO_RD]
- Write->Precharge to same sub-bank (to satisfy TIMING_CFG_1[WR_REC]

Impact: This only affects performance when using an 8-beat burst mode. An 8-beat burst mode is currently only supported for DDR1 memory when a 32-bit bus is used.

Workaround: Four-beat bursts could be used with DDR1 memory when using a 32-bit bus. However, the data ordering then is similar to the ordering used when DDR2 is used with a 32-bit bus and 4-beat bursts.

Fix plan: No plans to fix

DDR16: DDR controller self refresh

Description: When the DDR controller enters self-refresh and DDR_SDRAM_CFG[SREN] is set, the controller should issue a PRECHARGE_ALL command to the DRAMs if there are any open pages. If the PRECHARGE_ALL command is not issued, data could become corrupt when self refresh is entered. However, the DDR controller may not issue this PRECHARGE_ALL command when using a 16-bit bus. If a previous transaction has been sent to the controller that contains a memory select error, then the DDR controller skips the PRECHARGE_ALL command if it collides with the third or fourth read/write command of the transaction.

Impact: Data in memory could become corrupt if self refresh is entered without issuing the proper PRECHARGE_ALL commands.

Workaround: This bug is only present in 16-bit bus mode.

This bug may be present for transactions with memory select errors. It may also be present for all transactions if RD_TO_PRE is set to 1 or 2 cycles. This bug can be avoided for all transactions by programming DDRCDR[31], that is, the DRQ- Drain queue before sleep.

Fix plan: No plans to fix

DDR17: DDR controller fails after initialization

Description: The DDR I/O pins are in DDR1 mode by default after initial power up. To switch to DDR2 mode, the user must clear DDRCDR[DDR_cfg]. Once the value is written to this register it takes some time for the value to latch properly.

Impact: If enough time has not passed after writing to DDRCDR[DDR_cfg], the DDR2 controller may fail to initialize correctly.

Workaround: After writing to DDRCDR[DDR_cfg] allow 50 ms before continuing with the rest of the DDR controller initialization.

Fix plan: No plans to fix

DDR19: DDR MDQ/MDM output setup with regards to MDQS (t_{DDKHDS}) does not meet the specification at 333 MHz

Description: The DDR controller may not meet the minimum specification (800 ps) at 333-MHz MDQS for the MDQ/MDM output setup time (t_{DDKHDS}).

Impact: May lead to reduced output setup times.

Workaround: Use a setup time of 600 ps at 333 MHz.

Fix plan: Fixed in Rev 2.0

DDR21: MCK/$\overline{\text{MCK}}$ AC differential crosspoint voltage outside JEDEC specifications

Description: The crossover points of the MCK and $\overline{\text{MCK}}$ signals of the DDR controller are not meeting JESD79-2C specifications, which indicates that the crossover points should lie within ±125 mV range of reference voltage.

Impact: Disagreement with the JESD79-2C standard.

Workaround: To meet the JESD79-2C crosspoint voltage specifications, we recommend implementing the following connections between the DDR controller and DDR2 memory:

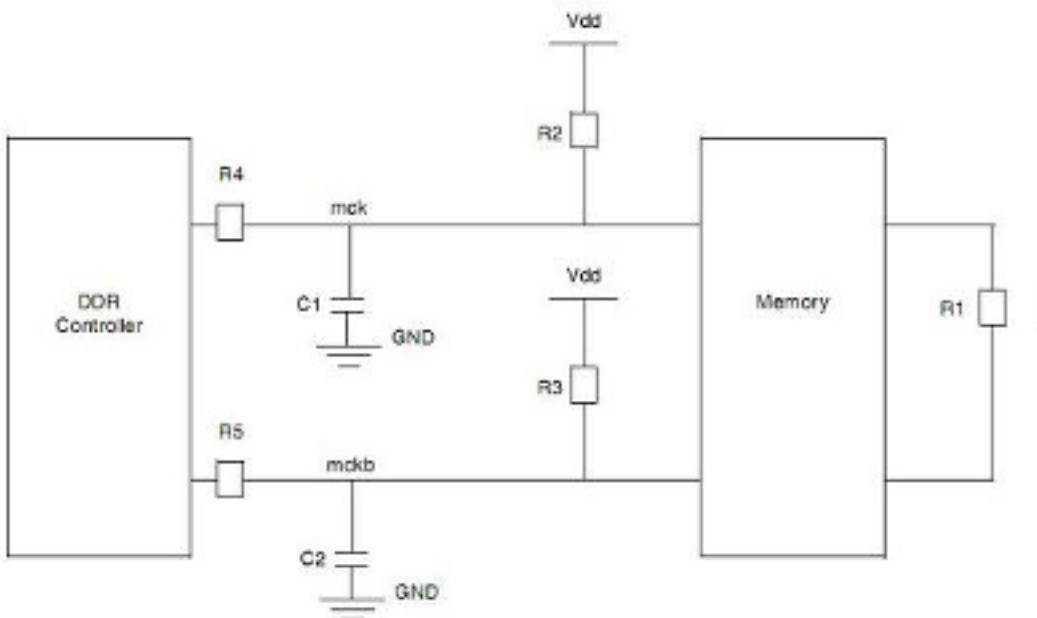

Figure 1. Recommended connections for DDR controller and memory

Effect of different circuit components on MCK and $\overline{\text{MCK}}$ signals:

1. Increasing R2 and R3 shifts signal levels up (used as pull up).
2. Increasing C1 increases rise/fall time of mck signal.
3. Increasing C2 increases rise/fall time of mckb signal.
4. Reducing R1 can help in shifting the signals up.
5. R4 and R5 can be used to make termination proper.

Component Placing:

R4, R5, C1, C2 near to SOC, R1, R2, and R3 have been placed at the end of clock transmission lines.

By considering these suggestions and choosing proper values of components for a particular board layout, one can keep crossover points within range.

Values used on the MPC8313ERDB reference design:

With these values we are able to get an acceptable waveform with crossover points within range.

R1 = 68 Ω

R2, R3 = Not used

C1, C2 = Not used

R4 = R5 = Not Used

Reference voltage = 1.8 V

We recommend varying termination resistor R1 first. Lowering R1 can help increase the voltage levels up. The only disadvantage is that it may add up more reflections. Therefore, the user must decide accordingly. If it does not give sufficient margin, the user can add the pull-up resistor (R3 and R4) option to further increase the voltage levels.

Exact values of components vary from design to design and some of them may not be required fpr all designs. We recommend that you make provisions for all these options in your design.

Fix plan: No plans to fix

DDR22: DRAMs using 12 × 8 × 2 configurations cannot be used in 16-bit bus mode

Description: DRAMs using 12 × 8 × 2 configurations cannot be used in 16-bit bus mode. The only JEDEC defined devices (×8 or ×16) that would violate this restriction are DDR1, 64 Mbit ×16 devices.

Impact: Data corruption will be seen in case of DDR 16-bit bus mode using 12 × 8 × 2 configuration

Workaround: Do not use 12 × 8 × 2 configuration for DDR 16-bit bus mode.

Fix plan: No plans to fix

DDR23: DDR read failure due to t_{DISKEW} spec violation

Description: When the minimum voltage swing (500 mV) on inputs specified by JEDEC spec is applied, there will be violation of t_{DISKEW} parameter.

Impact: DDR read operation is not guaranteed if peak-to-peak input voltage swing is less than 800 mV.

Workaround: Select the parallel termination resistor and configuration such that the peak-to-peak input voltage swing is more than 800 mV.

Fix plan: No plans to fix

DDR24: tDISKEW timing parameter may have insufficient margins

Description: During normal DDR operations, address phase of a read/write command can overlap with data phase of previous read command. In this scenario device may not meet t_{DISKEW} parameter with default signal swing for MDQ and MDQS.

Impact: DDR operations can fail due to insufficient tDISKEW margins.

Workaround: Depending on the revision of the silicon and the DDR type used, apply the relevant work around.

Note: D1 offset is IMMRBAR + DDR_OFFSET + 0xf00

For Silicon Rev 1.0 and 2.0:

- DDR1: Before memory controller is enabled:
- Select the RTT value of 85 Ω or higher for all data and strobe signals
- Set DDR_SDRAM_CFG[8_BE] and corresponding mode register to operate in 8-beat mode
- Set D1[16] to prevent pipelined operations in the DDR memory interface or clear CSB arbiter configuration register ACR[PIPE_DEP].
- Follow the DDR layout recommendations in Freescale Application Note AN2582, *Hardware and Layout Design Considerations for DDR Memory Interfaces*, to minimize all layouts related skews and noise.
- DDR2: Before memory controller is enabled:
- Set DDRCDR[ODT] to select 150 Ω ODT value.
- Clear DDR_SDRAM_MODE[14] to select DRAM driver strength to full.
- Set TIMING_CFG_1[CASLAT] = 4 clocks and TIMING_CFG_2[WR_LAT] = 3 clocks (note that the corresponding CASLAT setting in the DDR_SDRAM_MODE register should change to 4 clocks)
- Set D1[16] to prevent pipelined operations in the DDR memory interface or clear CSB arbiter configuration register ACR[PIPE_DEP].
- Follow the DDR layout recommendations in Freescale Application Notes AN2583, *Programming the PowerQUICC™ III/PowerQUICC™ II Pro DDR SDRAM Controller*, and AN2910, *Hardware and Layout Design Considerations for DDR2 SDRAM Memory Interfaces*, to minimize all layout related skews and noise.

For Silicon Rev 2.1:

- DDR1: Before memory controller is enabled:
- Select the RTT value of 85 Ω or higher for all data and strobe signals
- Clear DDR_SDRAM_MODE[14] to select DRAM driver strength to full
- Follow the DDR layout recommendations in Freescale Application Note AN2582, *Hardware and Layout Design Considerations for DDR Memory Interfaces*, to minimize all layouts related skews and noise.
- DDR2: Before memory controller is enabled:
- Set DDRCDR[ODT] to select 150 Ω ODT value
- Clear DDR_SDRAM_MODE[14] to select DRAM driver strength to full
- Follow the DDR layout recommendations in Freescale Application Notes AN2583, *Programming the PowerQUICC™ III/PowerQUICC™ II Pro DDR SDRAM Controller*, and AN2910, *Hardware and Layout Design Considerations for DDR2 SDRAM Memory Interfaces*, to minimize all layout related skews and noise.

Fix plan: No plans to fix

DDR25: DDR controller does not meet the t_{DDKHAX} min value at 333 MHz

Description: At 333 MHz, the minimum value specified by the hardware specifications is 2 ns. But the device has a minimum value of 1.9 ns.

Impact: DDR read operation is not guaranteed if peak-to-peak input voltage swing is less than 800 mV.

Workaround: Select the parallel termination resistor and configuration such that the peak-to-peak input voltage swing is more than 800 mV.

Fix plan: Fixed in Rev 2.1

eTSEC1: eTSEC reads to filer do not work

Description: When performing a memory-mapped read to the filer, eTSEC returns the data from the previous filer read, instead of the current one.

Impact: No support for simple reads to the filer.

Workaround: Perform two reads to the filer with the same address and use the data from the second read. There must be no intervening reads or writes to different addresses of the filer and the filer must be disabled to prevent hardware accesses.

Fix plan: Fixed in Rev 2.0

eTSEC2: eTSEC SGMII functionality is not available for any speed mode

Description: eTSEC SGMII functionality is not available for any speed mode (10/100/1000 Mbit/s).

Impact: SGMII functionality is not fully functional in MPC8313E, rev. 1.0.

Workaround: None

Fix plan: Fixed in Rev 2.0

eTSEC3: eTSEC Parser does not properly parse L3 fields

Description: The eTSEC parser does not properly process tunneled IP frames, resulting in loss of parser synchronization.

Impact: Tunneled IP frames received on the eTSEC Ethernet MAC and FIFO interfaces cannot be properly parsed and filed into receive queues.

Workaround: Do not enable parser recognition for L3 field, PRSDEP = 00 or 01 in Receive Control Register (RCTRL). Parsing and filling on L2 fields continues to be supported.

Fix plan: Fixed in Rev 2.0

Parsing of tunneled IP frames is disabled in Rev 2 silicon.

eTSEC5: WWR bit Anomaly

Description: DMACTRL[WWR] is intended to delay setting of IEVENT bits TXB, TXF, XFUN, LC, CRL, RXB, RXF until the system acknowledges that the buffer descriptor write data is actually in memory (L2 cache or DDR SDRAM), and not in flight in the system. There are certain cases when there are multiple outstanding BD updates, particularly in high latency memory scenarios, where an IEVENT can be lost when using DMACTRL[WWR] = 1.

Impact: If DMACTRL[WWR] = 1, then there is on occasion a missed IEVENT, or possibly an incorrect IEVENT assertion. This means that the interrupt could be missed altogether (BD still correctly updated in memory), or the IEVENT could be incorrect. In the case of it being incorrect, the IEVENT would not correspond to the BD at the head of the list, but would correspond to the BD second or third in the list.

Workaround: Set DMACTRL[WWR] = 0. The effect of setting DMACTRL[WWR] = 0 is that the interrupt may arrive at the processor before the update to the BD for the received packet that caused the interrupt has been completed in memory. This may or may not have any impact on the system depending on how packets are processed.

If the CPU reads the BD immediately after the interrupt, then in heavily congested systems it is possible that the CPU completes a read of the BD before the BD is closed by the eTSEC so that the BD's Empty bit is still set. In this case, software can either exit the packet processing routine and service the packet upon receiving the next interrupt, or it can schedule another interrupt to process the packet later.

Use of Rx interrupt coalescing of even a few packets reduce the chance of the CPU reading a BD whose update is still in flight to virtually zero, though it is still possible if multiple receive rings are in use.

Fix plan: Fixed in Rev 2.0

eTSEC7: RMCA, RBCA counters do not correctly count valid VLAN tagged frames

Description: According to the reference manual, RMCA increments for each multicast frame with valid CRC and a length between 64 and 1518 (non-VLAN tagged frames) or 1522 (single VLAN tagged frames) excluding broadcast frames. RBCA is the same definition except it counts broadcast frames and not multicast frames. The erratum is that for a valid VLAN tagged frame greater than 1518 the eTSEC does not increment these registers.

Impact: RBCA and RMCA do not increment for validly VLAN tagged Ethernet frames greater than 1518.

Workaround: There is currently no work around for counting these packets other than software running on the core.

Fix plan: Fixed in Rev 2.0

eTSEC8: RSTAT[RXF0] set regardless of destination ring if WWR=0

Description: If WWR=0, RSTAT[RXF0] may be set when a receive frame event occurs, even if the event actually occurs on a different RxBD ring.

Impact: Software cannot rely on RSTAT[RXF0] to indicate that a ring-0 receive-frame event occurred, or that receive-frame events on other RxBD rings will set the correct RSTAT[RXFn] bit.

Workaround: When RSTAT[RXF0] is set, software should check all active rings for the updated RxBD. If RSTAT[RXF1:RXF7] is set, only the corresponding ring needs to be checked.

See also eTSEC 5 (WWR Bit Anomaly) for a description of other software requirements when WWR=0.

Fix plan: Fixed in Rev 2.0

eTSEC9: Error in arbitrary extraction offset

Description: The byte offset for the arbitrary extraction filer feature is shifted such that the wrong bytes are extracted in some cases and some byte offsets cannot be extracted. The problem only applies to L2 extraction.

Impact: The following bytes cannot be extracted:

- With no VLAN/MPLS/SNAP/PPOE tag: Packet bytes 20-21 cannot be extracted.
- With 1 tag: Packet bytes 24-25 cannot be extracted.
- With 2 tags: Packet bytes 28-29 cannot be extracted.

Note that PPOE and SNAP count as two tags each.

L2 extraction of bytes other than the above requires software assistance as described in the workaround.

Workaround: Software must understand the shifting of bytes described below and compensate accordingly.

With one tag (VLAN/MPLS/PPOE):

- BxFFSET=0-7 extract preamble bytes 0-7.
- BxFFSET=8-27 extract bytes 0-19 of packet. Byte 0 is the first byte of the DA.
- BxFFSET=28-33 extract bytes 18-23 of packet.
- Beginning at offset 34, the pattern is criss-crossed within a 4-byte granularity and is repeated after every 4 bytes. For example:
  - BxFFSET=34 extract byte 28 of packet.
  - BxFFSET=35 extract byte 29 of packet.
  - BxFFSET=36 extract byte 26 of packet.
  - BxFFSET=37 extract byte 27 of packet.
  - BxFFSET=38 extract byte 32 of packet.
  - BxFFSET=39 extract byte 33 of packet.
  - BxFFSET=40 extract byte 30 of packet.
  - BxFFSET=41 extract byte 31 of packet.

With 2 tags (VLAN/MPLS/PPOE):

- BxFFSET=0-7 extract preamble bytes 0-7.
- BxFFSET=8-31 extract bytes 0-23 of packet. Byte 0 is the first byte of the DA.
- BxFFSET=32-37 extract bytes 18-27 of packet.
- Beginning at offset 38, the pattern is criss-crossed within a 4-byte granularity and is repeated after every 4 bytes. For example:
  - BxFFSET=38 extract byte 32 of packet.
  - BxFFSET=39 extract byte 33 of packet.
  - BxFFSET=40 extract byte 30 of packet.
  - BxFFSET=41 extract byte 31 of packet.
  - BxFFSET=42 extract byte 36 of packet.
  - BxFFSET=43 extract byte 37 of packet.
  - BxFFSET=44 extract byte 34 of packet.
  - BxFFSET=45 extract byte 35 of packet.

Fix plan: Fixed in Rev 2.0

eTSEC10: Limitations on the eTSEC1/eTSEC2 power supply connection

Description: The TSEC1_GTX_CLK125 signal is shared between eTSEC1 and eTSEC2. Internally TSEC1_GTX_CLK125 is supplied by LV_{DDB}, which supplies eTSEC1. So, when using RGMII/RTBI mode, the voltage level of TSEC1_GTX_CLK125 (supplied by an external PHY or oscillator) must be the same as the eTSEC1 voltage (controlled by SICRH[30]).

Impact: The eTSEC1 bank and TSEC1_GTX_CLK125 must operate at the same voltage level.

Workaround: The voltage level of TSEC1_GTX_CLK125 must be same as that of the eTSEC1 bank.

Fix plan: Fixed in Rev 2.0

eTSEC12: Tx IP and TCP/UDP Checksum Generation not supported for some Tx FCB offsets

Description: If the Tx FCB (Frame Control Block) 32-byte offset is 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E or 0x1F, IP and TCP/UDP header checksum generation do not function properly. The checksum value may be inserted in the wrong location or not inserted at all.

Impact: IP and TCP/UDP header checksum generation is not supported in LINUX and other systems in which headers are prepended to pre-aligned packet data, or where the alignment of the Tx FCB cannot be controlled.

This behavior applies to pseudo-header checksum insertion as well as checksum generation.

Workaround: Align Tx FCB to a 16 or 32-byte boundary.

If the alignment of TxFCB is not controllable, set TCTRL[TUCSEN]=0 and TCTRL[IPCSEN]=0 to disable IP and TCP/UDP header checksum generation.

Fix plan: Fixed in Rev 2.0

eTSEC13: 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

- First TxBD fetch for queue 0 with polling enabled (DMACTRL[WOP] = 0), or,
- Any fetch if EDIS[EBERRDIS] = 1
  - The Ethernet controller keeps refetching the same address, resulting in a livelock of the transmit or receive state machines. If the source of the error (for example, address mapping unit in the case or unmapped address, DDR controller in the case of ECC on memory data) has interrupts enabled for the error condition, the interrupt handler can resolve the error (by for example, mapping the unmapped address or writing the memory location with good ECC). The controller resumes normal function when it receives the fetch data without an error.
  - The controller can also recover from the livelock condition by toggling MACCFG1[TX_EN] for Tx livelock or MACCFG1[RX_EN] for Rx livelock.

Scenario 2

- First TxBD fetch for queue 0 with polling disabled
  - The Ethernet controller halts all Tx queues (TSTAT[THLTn] = 1, n = 0–7), but does not set IEVENT[EBERR]. Software can determine that the halt was due to an error rather than processing complete by examining the rings for ready TxBDs. EDIS[EBERRDIS] must be 0.

Scenario 3

- Tx data fetch
  - The Ethernet controller does not detect errors on Tx data fetches. IEVENT[EBERR] is not set for an address or data error on Tx data fetches, and the queues are not halted. The error is handled at the platform level, via an interrupt from the source of the error (for example, DDRC multibit ECC error or address mapping error).

Scenario 4

- Non-first TxBD fetch for queue 0, or,
- TxBD fetch for queues 1–7
  - The Ethernet controller sets IEVENT[EBERR] and halts all Tx queues (TSTAT[THLTn] = 1, n = 0 – 7]. This is the correct operation for Tx fetch error conditions. EDIS[EBERRDIS] must be 0.

Scenario 5

- RxBD fetch
  - The Ethernet controller sets IEVENT[EBERR] and halts the queue with the error (RSTAT[QHLTn] = 1). This is the correct operation for Rx fetch error conditions. EDIS[EBERRDIS] must be 0.

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

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

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

Workaround: All scenarios:

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

Transmit buffer descriptor work around:

Have software periodically check the state of the Tx queue halt bits. If a queue is halted, but still contains ready BDs, resolve any pending address or data error conditions before restarting the queues.

- **Option 1 for Tx queue 0:**

  Disable polling (set DMACTRL[WOP] = 1). Queue 0 error management then follows queue 1–7 error management (queue halt without error indicator, but no livelock).

- **Option 2 for Tx queue 0:**

  Enable all error interrupt enables for address and data errors in regions of memory used by TxBDs. Ensure error interrupt handlers resolve each error condition (unmapped address, uncorrectable ECC error, etc.) so the controller would eventually receive data without error and continue.

- **Option 3 for Tx queue 0:**

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

Fix plan: Partially fixed in Rev 2.0

Scenarios 1 and 2 fixed

Scenario 3 applies to all revisions of silicon. Scenarios 4 and 5 function properly as described above.

eTSEC14: RMON results may be unreliable

Description: The Tx MIB counter registers may contain arbitrary, incorrect values.

The affected registers are: TR64, TR127, TR255, TR511, TR1K, TMAX, TRMGV, TBYT, TPKT, TMCA, TBCA, TXPF, TDFR, TEDF, TSCL, TMCL, TLCL, TXCL, TNCL, TDRP, TJBR, TFCS, TXCF, TOVR, TUND, TFRG.

Impact: Tx RMON counter results may be unreliable. If the results are incorrect, they have arbitrary values.

Workaround: None, if affected.

Fix plan: Fixed in Rev 2.0

eTSEC15: Transmit jumbo frames greater than 2400 bytes may cause lost data, loss of BD synchronization, or false underrun error

Description: If the transmit processes a combination of up to four active frames which together exceed 9600 bytes, the Tx FIFO may overflow. When the TxFIFO overflows, one of several error conditions may occur. The scenarios below are representative, and may occur singly or in combination:

Scenario 1 (Lost data): The eTSEC overwrites part of a frame that has already started transmitting. The controller terminates the transmitting frame early without signaling an error condition or aborting the frame with bad CRC. In this scenario, the frame being loaded into the TxFIFO has TOE=1. [original eTSEC-55]

Scenario 2 (Lost BD synchronization): The eTSEC overwrites part of a frame that has already started transmitting. The controller transmits parts of two frames as a single frame with good CRC. Only the first frame's BD is closed. As each subsequent frame is transmitted, the BD of the previous frame is closed. The controller never recovers synchronization of BD to transmitted frame. This can occur with TOE=1 or TOE=0.

Scenario 3 (False underrun error): The eTSEC overwrites part of a frame that has already started transmitting. The controller terminates the transmitting frame with invalid CRC and halts (TSTAT[THLTn]=1). In addition, a transmit underrun error is falsely reported (IEVENT[XFUN]=1 and TxBD[UN]=1). This can occur with TOE=1 or TOE=0.

Impact: Combinations of frames that include jumbo frames greater than 2400 bytes may cause lost data, lost frames or false underrun indication in systems where the transmit throughput can fall behind the memory fetch throughput. This can occur with a fast memory subsystem, a slow interface, or collisions on the interface.

Workaround: Option 1: Limit jumbo frames to 2400 bytes maximum size on transmit.

Option 2: If using jumbo frames larger than 2400 bytes, limit the active TxBDs so no combination of up to four frames exceeds 9600 bytes.

Fix plan: Fixed in Rev 2.0

eTSEC16: Parsing of tunneled IP packets not supported

Description: Encapsulation of IP in IP in either TCP or UDP packets is not supported by eTSEC parser. This applies to both IPv4 and IPv6.

A tunneled IP packet is an IP/TCP or IP/UDP packet and one of the following:

1. IPv4 header with a value of either 4 or 41 in the Protocol field, indicating that the next header is either another IPv4 header or IPv6 header, respectively
2. IPv6 header with a value of either 4 or 41 in the Next Header field, indicating that the next header is either a IPv4 header or another IPv6 header, respectively

When the parser encounters a tunneled IP packet, it terminates its parsing operation at the end of the outer IP header.

Impact: Because the parser terminates its parsing operation, only the outer IP header checksum, if it exists, is checked. Checksums of additional L3 headers within the packet are not checked, and no TCP/UDP checksums are checked.

Workaround: If L3 or L4 parsing is enabled and tunneled packets are expected, software must examine each packet header to see if it is a tunneled IP packet. If the packet is a tunneled IP packet, software should calculate and check all checksums other than those in the outer IP header.

Fix plan: No plans to fix

Disable parsing of tunneled IP packets in revision 2 silicon.

eTSEC18: Parsing of MPLS label stack or non-IPv4/IPv6 label not supported

Description: The parser does not continue parsing beyond multi-label stack, or MPLS frame with a label other than IPv4 or IPv6. The RxFCB is written as 0x0000_00ff_0000_0000 (no layer 3 header recognized).

Impact: The eTSEC cannot parse beyond an MPLS stack of greater than depth 1. It also cannot parse beyond an MPLS header with label other than IPv4 or IPv6.

Workaround: Limit MPLS Ether-type packets to MPLS label stack depth = 1 with IPv4 or IPv6 label.

Fix plan: Fixed in Rev 2.0

eTSEC19: Compound filer rules do not roll back the mask

Description: The eTSEC filer has associated with it a mask value that is used when rules are comparing fields of the packet against properties in the RQFPR table. The mask sets “don’t cares” in the comparison. When building a compound rule through the use of the AND bit either in or outside of a cluster guard rule (CLE = 1) you can set masks as appropriate for the subsequent rule by setting CMP = 00/01, PID = 0, and RQPROP = “desired mask.” If however the chained rule fails for any single rule, the mask should revert back to what it was prior to entering the rule chain. The erratum is that the mask does not roll back and the resulting mask can be unknown.

Impact: Some rules may falsely match or not match causing the filing of a frame to the wrong queue or incorrectly rejecting the frame in the case of an assumption of the mask being a certain value.

Workaround: When using a compound rule that consists of SETMASK rules, the user must put another SETMASK rule after the last rule in the chain that resets the mask to the value it was prior to entering the chain. The following table shows a compound rule example for work arounds.

Table 5. Compound Rule Example for Work Arounds

Table Entry	RQCTRL CLE	REJ	AND	Q	CMP	PID	RQPROP	Comment
0	0	0	1	0	0	7	0x0000_0800	—
1	0	0	1	0	0	0	0xFFFF_0000	Setmask
2	0	0	1	0	0	12	0xC054_1200	—
3	0	0	1	0	0	0	0xFF00_0000	Setmask
4	0	0	0	5	0	13	0xC055_0000	—
5	0	0	0	0	0	0	0xFFFF_FFFF	Work around

Fix plan: Fixed in Rev 2.0

eTSEC20: Filer does not support matching against broadcast address flag PID1[EBC]

Description: The controller clears its copy of the Ethernet broadcast address before extracting filer properties, so the filer cannot correctly match based on broadcast address (PID1[EBC]). The frame itself is not affected.

Impact: If broadcast address matching is enabled, frames may be incorrectly filed or rejected.

Workaround: Mask off matching on broadcast address flag (PID1[EBC] = 1) by clearing the bit 16 of the mask_register. If the rule needs to be able to distinguish broadcast addresses as defined by IEEE Std 802.3™-2005 is all 1's in the destination address field, then use a filer rule with PID3 and PID4 (destination MAC address) to match on broadcast Ethernet frames.

Fix plan: Fixed in Rev 2.0

eTSEC21: L3 fragment frame files on non-existent source/destination ports

Description: If the controller detects a L3 fragment, it should terminate parsing. Instead, it continues to the end of the header looking for a L4 header, extracts non-existent source and destination ports, and may file the fragment based on port match.

Impact: L3 fragment frames may be parsed and filed incorrectly.

Workaround:
- Option 1: Include a filer rule to reject on PID1[IPF] at the beginning of the table.
- Option 2: Limit parsing to L2 by setting RCTRL[PRSDEP] = 01. Note that limiting parsing to L3 by setting RCTRL[PRSDEP] = 10 is not a valid work around .

Fix plan: Fixed in Rev 2.0

eTSEC22: RxBD[TR] not asserted during truncation when last 4 bytes match CRC

Description: The eTSEC truncates any receive frame larger than MAXFRM, unless Huge Frame Enable is set (MACCFG2[Huge Frame] = 1). The proper behavior for the controller is to set the RxBD[TR] bit (and RxBD[LG], if RxBD[L] = 1) for any truncated frame. If the last 4 data bytes received before truncation happens to match the running CRC, then RxBD[TR] (and RxBD[LG]) is not set even though the frame has been truncated.

Impact: If the 4 data bytes just before MAXFRM bytes into the frame match the running CRC for the frame, the packet is silently truncated (no error indication via RxBD[TR]).

Workaround: None

Fix plan: Fixed in Rev 2.0

eTSEC24: Parser results may be lost if TCP/UDP checksum checking is enabled

Description: When the parser is enabled and RCTRL[TUCSEN]=1, if the first RxBD data arrives from memory the same cycle that parsing of the packet completes, all the fields of the RxFCB except the receive queue index will be written with zeroes instead of the parser results.

Impact: When a single-cycle collision of first RxBD prefetch and parsing complete occurs, the parser results other than receive queue index are lost and the VLN, IP, IP6, TUP, CIP, CTU, EIP, ETU, PERR, PRO, VLCTL bits of the RxFCB are set to all zeroes.

If VLAN extraction is enabled (RCTRL[VLEX]), the VLAN ID is lost.

Workaround: Option 1: Disable TCP/UDP checksum checking by setting RCTRL[TUCSEN]=0.

Option 2: Disable VLAN extraction by setting RCTRL[VLEX] and check the contents of the RxFCB. If the contents are zero, replicate the parser algorithm in software to determine the correct parser results.

Fix plan: Fixed in Rev 2.0

eTSEC25: Transmission of truncated frames may cause hang or lost data

Description: If all three of the following conditions are concurrently met the controller may hang, or drop some bytes from the second frame without any error indication:

1. The Ethernet controller truncates a transmitted frame which is larger than MAXFRM
2. The following frame has TOE = 1
3. The two frames together are large enough to fill the 10-Kbyte Tx FIFO without truncation

Impact: Truncating frames larger than MAXFRM may cause a transmit hang or lost data if combined with TOE = 1 frames.

Workaround:
- Option 1: Disable truncation by setting MACCFG2[Huge Frame] = 1.
- Option 2: Turn off TCP/IP offload enable by setting TxBD[TOE] = 0.

Fix plan: Fixed in Rev 2.0

eTSEC27: Transmitting PAUSE flow control frame may cause transmit lockup

Description: The process of generating a PAUSE control frame may cause the controller to stop transmitting (Tx lockup). After the controller enters a Tx lockup state, only a reset of the eTSEC and associated software allow it to start transmitting frames again. Pause flow control frame generation can be triggered by reaching the Rx FIFO threshold (basic flow control: MACCFG1[Tx_Flow]), running out of Rx buffer descriptors (lossless flow control: RCTRL[LFC]), or direct software control (TCTRL[TFC_PAUSE]).

Impact: Transmit flow control, lossless flow control, and software generation of pause flow control frames may cause the Ethernet controller to stop transmitting and require a reset of the controller.

Workaround: Disable transmit flow control by setting MACCFG1[Tx_flow]=0.

Fix plan: Fixed in Rev 2.0

eTSEC28: eTSEC does not support parsing of LLC/SNAP/VLAN packets

Description: eTSEC supports parsing of LLC/SNAP headers following the Ethernet 802.3 length field interpretation. It also supports 802.1p VLAN tags. However, it does not support the encapsulation defined as Ethernet length, followed by LLC/SNAP, followed by VLAN. The parser prematurely completes “normally” upon encountering the VLAN Ether-type at the end of the LLC/SNAP encoding. The erratum is that no layer 3, layer 4, or VLAN information is submitted to the filer or reported in the RxFCB.

Impact: This unique packet type is not parsed beyond layer 2, including any VLAN processing, because the parser terminated before the VLAN tag was found.

Workaround: Software running on the host has to parse these packets because they indicate no parsing functions performed by eTSEC.

Fix plan: Fixed in Rev 2.0

eTSEC29: Arbitrary extraction on short frames uses data from previous frame

Description: If the Ethernet controller receives a frame which is smaller than one of the defined offsets for arbitrary extraction (RBIFX), it should set the corresponding byte of the ARB property sent to the filer to 0. Instead it returns the corresponding byte extracted from the previous frame.

Impact: If filing based on arbitrary extraction of bytes, frames shorter than the byte offset may be improperly filed: filed to the wrong queue, rejected when they should be accepted, or accepted when they should be rejected.

Workaround: Option 1: Use only RBIFX[BnCTL]=01 (extract from offset of DA-8 bytes). For valid Ethernet frames (minimum length 64 bytes), the 6-bit offset cannot go beyond the end of the frame.

Option 2: Do not use the ARB filer property to reject frames if the controller may receive frames shorter than the location of any arbitrary extraction byte offset. Software must handle short frames which may be filed in the wrong queue

Fix plan: Fixed in Rev 2.0

eTSEC30: Preamble-only with error causes false CR error on next frame

Description: If a receive error (RX_ER = 1) occurs on an Ethernet preamble, which is truncated before start of frame, the error indicator is carried over until the next start of frame is received and reported on the following frame by setting RxBD[CR] = 1.

Impact: A preamble-only sequence with error causes a false CR (code group or CRC) error on the next frame.

Workaround: System software is aware that a frame is preamble, it can force RX_ER deasserted whenever this kind of frame sent over to eTSEC, or it can not sent any frame to eTSEC with preamble.

Fix plan: Fixed in Rev 2.0

eTSEC31: eTSEC filer reports incorrect Ether-types with certain MPLS frames

Description: The eTSEC filer gets a property under PID = 7 called ETY. This usually corresponds to the last Ether-type that was encountered in a packet as it was parsed. In the case that there is an MPLS label in the packet, then the Ether-type is incorrectly returned as the last 2 bytes of the MPLS label. The last 2 bytes correspond to the LSB 4 bits of the label, the EXP field, the S field, and the TTL field. The eTSEC does not know of any header types that follow other than IPv4 or IPv6 through the use of reserved label values “IPv4 Explicit Null” and “IPv6 Explicit Null.” Hence the Ether-type is always left as the last 2 bytes of the MPLS label.

Impact: MPLS tagged packets report the incorrect Ether-type (8847 for MPLS unicast or 8848 for MPLS multicast).

Workaround: Use arbitrary extraction bytes to compare to the actual Ether-type if a filer rule is intending to file based on an MPLS label existence.

Fix plan: Fixed in Rev 2.0

eTSEC33: Parser does not check VER/TYPE of PPPoE packets

Description: The Ethernet controller supports PPPoE VER/TYPE = 1 packets. For PPPoE packets with VER/TYPE not equal to 1, the controller should stop Ethernet parsing and treat it as an unrecognized PPPoE packet. Instead, the controller does not check the VER/TYPE field, and assumes it is 1.

Impact: PPPoE packets with VER/TYPE that are not type 1 are parsed as if they are type 1 PPPoE.

Workaround: None

Fix plan: Fixed in Rev 2.0

eTSEC34: Some combinations of Tx packets may trigger a false Data Parity Error (DPE)

Description: Some combinations of Tx packets may incorrectly generate parity when loading the Tx FIFO. When the data is unloaded from the FIFO to be transmitted, if parity detection is enabled (EDIS[DPEDIS] = 0), a false parity error is flagged (IEVENT[DPE]).

The packet combinations that trigger the error are as follows:

1. An initial frame (X) which is not a multiple of 8 bytes in size, and
2. A subsequent shorter frame (Y) which is not a multiple of 8 bytes in size, and
3. A specific relationship between TxFIFO write pointer used for frame (Y) relative to frame (X) just prior to EOF (which cannot be controlled by the user), and
4. A data dependency between frames (X) and (Y) - on average only 50% of the scenarios matching (1)–(3) result in a false DPE being reported.

Impact: Systems which transmit packet combinations that trigger this false parity error may see some performance degradation due to false parity error interrupt service processing. No actual data corruption occurs as a result of the false parity error.

Workaround: Although it is possible to prevent false parity error indications by disabling SRAM parity detection (accomplished by setting EDIS[DPEDIS] = 1), this can have the undesirable effect of masking indications of real parity errors. Unless the specific application is experiencing a significant number of false parity errors that are resulting in unsatisfactory performance degradation, it is recommended that SRAM parity detection remain enabled. Please refer to eTSEC 38 for more information.

Fix plan: Fixed in Rev 2.0

eTSEC35: Back-to-back Rx frames may lose parser results of second frame

Description: In some circumstances, the parser results for a frame may be calculated when the controller is still processing the previous frame. If this scenario occurs, the parser results for the second frame are discarded, and the preloaded all zeroes value of RxFCB is returned instead. The circumstances are related to format of the two frames, but are not controllable by the receiver or the user.

Impact: Under some conditions that are not controllable by the receiver or user, parser results for the second of a back-to-back frame may be lost. If VLAN extraction is enabled, the VLAN ID is lost.

Workaround: Option 1: Disable all parsing functions (RCTRL[PRSDEP] = 00.

Option 2: Because this erratum impacts the RxFCB only, the filer still gets the correct information and be able to file to the appropriate queue or reject. If software now finds RxFCB all zeros, it must assume that it encountered this error. The only other time the RxFCB is all zeros is when the eTSEC didn't find any L2–L4 information that is recognized.

Fix plan: Fixed in Rev 2.0

eTSEC36: Generation of Ethernet pause frames may cause Tx lockup and false BD close

Description: The process of generating a PAUSE control frame may cause the controller to transmit two pause frames instead of one. If this occurs, the controller erroneously sees the second pause frame as a transmitted data frame, closes the next TxBD and decrements the running counter of frames in the TxFIFO.

Pause flow control frame generation can be triggered by reaching the Rx FIFO threshold (basic flow control: MACCFG1[Tx_Flow]), running out of Rx buffer descriptors (lossless flow control: RCTRL[LFC]), or direct software control (TCTRL[TFC_PAUSE]).

Impact: Transmit flow control, lossless flow control, and software generation of pause flow control frames may cause the Ethernet controller to stop transmitting and falsely close a TxBD for an un-transmitted frame.

Workaround: Disable transmit flow control by setting MACCFG1[Tx_flow] = 0.

Fix plan: Fixed in Rev 2.0

eTSEC37: eTSEC half duplex receiver packet corruption

Description: When eTSEC is configured to run in half-duplex mode, it relies on the PHY to isolate its receiver from receive data during packet transmission. The PHY indicates contention on the wire via the COL signal in xMII, which forces the eTSEC into the standard backoff algorithm. If the eTSEC does in fact get receive data (RX_DV = 1) while it is transmitting (TX_EN = 1), it receives the packet data corrupted and inflected from its original size on its way to memory.

This issue likely arises from the transmit data being wrapped at the PHY and send to the eTSEC receiver. This issue impacts running internal and external loopback while the eTSEC is configured for half-duplex operation.

Impact: Receive data corruption occurs during simultaneous packet transmission and reception in half-duplex. Additionally, the corruption ends up with various receive MIB counters counting invalid errors depending upon the original packet size and the type of corruption (RFCS, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR). Also, the receive byte counters indicate packets larger than those transmitted were received.

Workaround: The eTSEC can be configured through the filer to discard such packets that are wrapped externally in this manner through the use of MAC addresses match and drop. The receive MIB counters still count the packets as errored when they were in fact not errored.

If loopback mode is desired, the eTSEC must be configured for full-duplex operation.

Fix plan: Fixed in Rev 2.0

eTSEC38: eTSEC Data Parity Error (DPE) does not abort transmit frames

Description: eTSEC supports parity protection on its TX FIFO to protect against memory errors of two types: soft errors and hard errors. Soft errors can be caused by a RAM bit cell temporarily losing its value due to an event such as an alpha particles or neutron impact, but it holds the next write. Hard errors can be caused by a RAM bit cell no longer reliably holding a 0 and/or 1 written to it.

In either case, the parity error indicates that either at least one bit in a 16-bit word of the frame data to be transmitted is incorrect, or that the parity bit itself is incorrect. The correct behavior is to make sure that the peer on the other end of the link or connection is notified of this data corruption. This can by done by making sure that FCS at the end of the frame is incorrect, asserting TX_ER, or, in 16-bit FIFO encoded mode, sending an error code group. The controller does assert a local error event (IEVENT[DPE]), but does not give the link peer any error indication.

Impact: If a parity error occurs on a data bit, the corrupted packet is transmitted masked as a good packet with good FCS.

Higher layer protocols that have additional error checking such as TCP/UDP checksums may detect the corrupted data, depending on the location of the bad bit.

Workaround: None

Fix plan: Fixed in Rev 2.0

eTSEC39: May drop Rx packets in non-FIFO modes with lossless flow control enabled

Description: Lossless Flow Control (enabled by setting RCTRL[LFC] = 1, is intended to ensure zero packet loss of receive packets due to lack of RxBDs. This is done by applying back pressure when the number of free RxBDs drops below a critical threshold set by software. In FIFO modes (both GMII-style and encoded), the back pressure is applied by asserting receive flow control (CRS pin) until the number of RxBDs exceeds the critical threshold.

In non-FIFO modes, the back pressure is applied by transmitting a pause control frame with the pause time as in PTV[PT]. If the number of free RxBDs is still below the critical threshold when half the pause time has expired, the controller sends another pause frame and resets the counter.

If another pause frame is transmitted during the critical window when the 1/2 PTV pause counter expires, either due to software setting TCTRL[TFC_PAUSE] or through basic flow control when the data in the RxFIFO exceeds its threshold, then the 1/2 PTV pause counter may stop counting and the state machine may cease generating automated pause extensions. If the pause time associated with the last pause control frame expires with the number of free BDs still below the critical threshold, then frames may be dropped with the IEVENT[BSY] bit set indicating frame loss due to lack of buffer descriptors.

Only the transmission of another pause frame due to TCTRL[TFC_PAUSE] or data in RxFIFO exceeding threshold restarts the 1/2 PTV counter and state machine for lossless flow control.

Impact: The Ethernet controller may run out of RxBDs and drop receive packets even if lossless flow control is enabled, in a MAC mode (GMII, RGMII, SGMII, MII, RMII, TBI, RTBI).

Workaround:

- **Option 1:**

  Enable interrupts on transmit of pause frames (IMASK[TXCEN] = 1). For every interrupt due to pause frame transmission (IEVENT[TXC] = 1), enable a counter such as a cycle count in the Performance Monitor block which would overflow and generate an interrupt after 2/3 of PVT time. If the counter is already enabled, reset the count to 2/3 of PTV time. In the overflow event interrupt handler, check the current number of free receive buffer descriptors versus the threshold. If the number of free BDs are below the threshold, manually transmit a pause frame by setting TCTRL[TFC_PAUSE] = 1. This also restarts the lossless flow control state machine. In either case (free BDs above or below threshold), disable the counter until the next transmit control frame event.

- **Option 2:**

  Enable interrupts on dropped packets due to lack of RxBDs (EDIS[BSYDIS] = 0, IMASK[BSYEN] = 1). On detecting a busy dropped packet event (IEVENT[BSY] = 1), manually transmit a pause frame by setting TCTRL[TFC_PAUSE] to restart the lossless flow control state machine.

NOTE

The probability of packet loss in MAC modes can be reduced by increasing the pause time value in PTV[PT], or increasing the critical RxBD threshold in RQPRMn[PBTHR], or both.

Fix plan: Fixed in Rev 2.0

eTSEC40: Rx synchronization error may cause corrupted packets and limitations with Gigabit operation

Description: The receiver logic of the Ethernet controller synchronizes indications of events as they pass across several clock domains. At low platform frequencies, the controller may fail to capture a pulse and miss the indication of the event. One of these event indications controls writing to the Rx FIFO. If the event indication is missed, several scenarios could occur:

- Two or more accepted frames could be concatenated together. The first frame is corrupted (missing some bytes at the end of the frame). The second frame is complete in memory. Only the last frame's length is reported in the BD, not the combined lengths.
- A frame which should be accepted may be partially flushed to memory without closing the last BD. The BD stays open until a new frame is accepted.
- For three or more successive frames, where the first and last frames are accepted and the middle one(s) is/are rejected, the rejected frame(s) may be all concatenated together with the following accepted frame and written to memory.

Impact: An Rx synchronization error could cause packets to be corrupted, with multiple packets concatenated, rejected frames partially or fully written to memory, and/or an RxBD left open which should be closed (loss of BD synchronization).

Workaround: Run the Ethernet controller at a CSB to RX interface clock at a ratio of at least 1.3:1 to ensure the event can be correctly sampled. In the case of a Gigabit interface using a 125MHz clock, the CSB must be running at least 162.5 MHz. Therefore, customers with a 266-MHz core frequency device cannot run the eTSECs at Gigabit speed. Additionally, the TX interface clock must be run at the same frequency as the RX interface clock.

Fix plan: Fixed in Rev 2.0

eTSEC41: Multiple BD Tx frame may cause hang

Description: In section 15.6.7.2 of the MPC8313ERM (Rev. 1), it states:

"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 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: Fixed in Rev 2.0 if multiple Tx queues enabled.

No plans to fix for single queue case.

eTSEC42: 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

eTSEC43: eTSEC RGMII data to clock output skew at transmitter (t_{SKRGT}) does not meet the specification

Description: eTSEC RGMII controller may not meet maximum specification ±500 ps) for data to clock output skew at transmitter (t_{SKRGT}).

Impact: RGMII 1000 speed operation.

Workaround: As per RGMII specs data skew should be 1.0 ns to 2.6 ns at the receiver end. TSEC1/2_GTX_CLK needs to delayed either on the PCB or PHY to get a data skew of 1.0–2.6 ns at the receiver end. Bits SICRH[28] or SICRH[29] can be set to get the data skew required at the receiver end, by enabling delay on TSEC1_GTX_CLK or TSEC2_GTX_CLK, respectively. Delay need not be added on either the PCB or PHY.

Fix plan: Fixed in Rev 2.0

eTSEC44: No parser error for packets containing invalid IPv6 routing header packet

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

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

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

For Rev 1.0, the logic would continue to parse the invalid packet normally as if there is no error in the packet. CTU and ETU bits work as if the packet is not invalid.

For Rev 2.0 and later, the logic will do the following:

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

Impact: An IPv6 routing header with segments left greater than number of DAs is not flagged as an invalid packet.

Workaround: For Rev 1.0:

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.

For Rev 2.0 and later:

Option 1 - Set up the filer to detect the invalid IPv6 routing header using PID 1's bit 11.

Option 2 - Apply the same workaround as in Rev 1.0 only if CTU bit is 0.

Fix plan: Partially fixed in Rev 2.0

eTSEC45: eTSEC parser does not perform length integrity checks

Description: The eTSEC currently only uses the total length reported in the IPv4 or IPv6 header when calculating checksums. This checksum calculation includes the length used in the pseudoheader, and the actual length of the data in the payload. Proper operation when parsing a subsequent L4 header that has a length field (be it payload and/or header) should be to check for consistency against what is reported in the outer IP header. If there is a mismatch, the eTSEC should signal a parse error and not perform the UDP or TCP payload checksum check (for example, RxFCB[PERR] = 10 and RxFCB[CTU] = 0).

One simple example of this is that UDP has its own payload length. If eTSEC encounters a simple IPv4/UDP packet it should take the IP total length field, subtract IP header length and that should equal the UDP payload length. If it doesn't then this packet is malformed.

Impact: Could get false checksum failures or false checksum passes.

Workaround: False checksum fails can be worked around by rechecking them in software running on the host.

Fix plan: Fixed in Rev 2.0

eTSEC46: eTSEC does not verify IPv6 routing header type field

Description: The RFC2460 (current referenced standard for IPv6 operation) states that when encountering a packet with an unrecognized Routing Type Value, and the field “segments left” is non-zero, the node must discard the packet and return an ICMP Parameter problem, Code 0, message to the packet’s source address. The eTSEC only recognizes type0 routing headers, but incorrectly interprets all Routing Type fields as type0 (for example, ignores the type field and continues parsing the packet including upper layer protocol checksums). The correct behavior is to signal parser error, and not check upper layer checksums

Impact: eTSEC parser/checksum engine incorrectly interprets non-type0 IPv6 routing headers. Functionally, this is a future-proof issue because there are currently no other type-fields defined.

Workaround: If this device is operating in a network that is using non-type0 IPv6 routing headers, then the upper layer processing (IPv6 extension headers, and payload checksumming operations) must be performed in software.

Fix plan: Fixed in Rev 2.0

eTSEC47: No parse after back-to-back IPv6 routing header

Description: Upon encountering back to back IPv6 routing extension headers following an IPv6 header, the eTSEC stops further parsing, and place 0xFF in the RxFCB[PRO], and RQFPR,pid = 0xB[L4P]. If there are 2 or more IPv6 routing extension headers, and there is either an IPv6 hop-by-hop extension header (see reference to RFC below) or an IPv6 destination options header or both between the routing headers, then the eTSEC continues to parse the sequence.

According the RFC2460, “Each extension header should occur at most once, except for the Destination Options header which should occur at most twice (once before a Routing header and once before the upper-layer header).” However, it goes on further to state, “IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, except for the Hop-by-Hop Options header which is restricted to appear immediately after an IPv6 header only.”

Impact: Upon encountering a packet with 2 or more IPv6 routing extension headers that are back to back, the eTSEC indicates RxFCB[IP] = 1, RxFCB[IP6] = 1, and RxFCB[PRO] = 0xFF. All layer 4 related information is zero (TUP, CIP, CTU, ETU), even if there is a recognizable L4 protocol field following the extension headers.

Workaround: Software must parse the L4 information out of packets that indicate PRO = 0xFF.

Fix plan: Fixed in Rev 2.0

eTSEC48: L4 info passed to filer in L2/L3-only mode

Description: RCTRL[PRSDEP] has parse control for L2 and L3 (10) and L2, L3 and L4 (11). In the case of L2 and L3, the eTSEC goes ahead and continues to parse into L4 protocols and update both RxFCB and filer PID = 1 fields for TCP and UDP.

Impact: The L4 protocols are parsed and status bits are set even when the eTSEC is not programmed to include L4 parsing.

Workaround: User can simply ignore the bits associated with L4 protocols. In the case of filer PID = 1, user must mask bits associated with TCP and UDP.

Fix plan: Fixed in Rev 2.0

eTSEC49: MAC header data corruption

Description: eTSEC transmit data corruption can occur when using a configuration that includes all the following features:

- eTSEC system clock < 167 MHz which corresponds to 167-MHz platform in the MPC8313ESC8311E.
- MACCFG2[PAD/CRC] = 0 and MACCFG2[CRC En] = 0 and TxBD[TC] = 0 and TxBD[Pad/CRC] = 0
- TxBD[TOE] = 1

The corruption appears in the form that periodically the first 8 bytes of a frame are the same as or merged with the last 8 bytes of the previous frame. This means that the MAC destination address and 2 bytes of the MAC source address are corrupted. Because this issue requires that the eTSEC not append the CRC to appear, the error shows up as a CRC error at the other end of the link, along with the MAC address corruption.

Impact: First 8 bytes of a frame may be corrupted if CRC append is disabled and TxTOE is enabled.

Workaround: Option 1: Run the platform clock >= 167 MHz.

Option 2: Enable MAC-generated CRC append by setting MACCFG2[PAD/CRC] = 1 or MACCFG2[CRC En] = 1.

Option 3: Turn off Tx TOE function by setting TxBD[TOE] = 0.

Fix plan: Fixed in Rev 2.0

eTSEC50: Rx_en synchronization may cause unrecoverable Rx errors at startup

Description: There is no synchronization between MACCFG1[rx_en] and all rx logic that is clocked by platform clock. The rx_en signal is driven by the interface clock (tx_clk) thus it is necessary for it to be synchronized with the platform clock before driving any platform logic. The problem is that this signal is not currently synchronized therefore it is possible that this signal may create a harmful glitch causing platform clock driven latches to not come out of reset properly. This problem may only be manifested when there is a 0->1 transition of MACCFG1[rx_en] which typically is driven by configuration software coming out of reset.

Impact: There is a very small chance that certain rx platform clock driven latches may not be properly retaining their reset state if a glitch is occurring when MACCFG1[rx_en] is changing from 0 to 1. However, the fail is probably highly silicon dependent. Some may have it and others may not have it at all.

Workaround: There is no good work around for this issue. However, it may be possible to set up loop back mode and send a packet to ensure that the logic is functionally properly coming out of reset. If the logic is not functioning during loop back, it may be necessary to toggle MACCFG1[rx_en] one more time and reapply the loop back test.

Fix plan: Fixed in Rev 2.0

eTSEC54: Frames greater than 9600 bytes with TOE = 1 will hang controller

Description: The eTSEC supports frames up to 9600 bytes (huge or jumbo frame). If a frame has TOE = 1, it must be no more than 9600 bytes to fit entirely into the Tx FIFO. If the frame is larger, the controller hangs, because it must have the last byte of data in the FIFO to calculate the checksum and allow the frame to start transmission.

Impact: A frame larger than 9600 bytes with TOE = 1 hangs the Ethernet controller.

Workaround: For frames larger than 9600 bytes, set TxBD[TOE] = 0. For frames with TxBD[TOE] = 1, ensure Tx frame length ≤ 9600 bytes.

Fix plan: Fixed in Rev 2.0

eTSEC55: Arbitrary Extraction cannot extract last data bytes of frame

Description: If the arbitrary extraction offset defined in the RBIFX register points to data in the last beat of a frame, the associated ARB property sent to the filer may be zero instead of the data at the designated offset, depending on packet type and length.

The following packet and extraction types are affected:

- L2, L3 or L4 extraction of packets with frame length 4n or 4n + 3
- L4 extraction of TCP/UDP packets with IP total length 4n + 1, 4n + 2, or 4n + 3.

Impact: The following conditions apply to any type of frame and L2, L3 or L4 extraction:

- For frame length of 4n, the last 2 bytes of the frame are not extractable. This applies to L2, L3 or L4 extraction in MAC or FIFO modes.
- For frame length of 4n + 3, the last 1 byte of the frame is not extractable. This applies to L2, L3 or L4 extraction in MAC or FIFO modes.

The following conditions apply to L4 extraction from a packet with TCP/UDP data (when RCTRL[PRSDEP] = 11, RCTRL[TUCSEN] = 1):

- For IP total length of 4n + 1, the L4 byte offsets 4n + m - <IP header length> are not extractable, for m = 1, 2, or 3.
- For IP total length of 4n + 2, the L4 byte offsets 4n + m - <IP header length> are not extractable, for m = 2 or 3.
- For IP total length of 4n + 3, the L4 byte offset 4n + 3 - <IP header length> is not extractable

Workaround: None

Fix plan: Fixed in Rev 2.0

eTSEC56: Setting RCTRL[LFC] = 0 may not immediately disable LFC

Description: Lossless flow control is controlled by RCTRL[LFC]. Setting RCTRL[LFC] = 0 should immediately disable the lossless flow control state machine and stop the sending of pause frames based on number of free RxBDs. The controller instead waits until the state machine is idle before disabling it. If the state machine has been triggered by the number of free RxBDs falling below the threshold, the controller continues sending pause frame extensions until the number of free RxBDs exceeds the threshold.

Impact: Generation of pause frames due to lack of free RxBDs may continue for a time after setting RCTRL[LFC] = 0.

Workaround: When disabling LFC, first set RCTRL[LFC] = 0, then poll the number of free RxBDs until it exceeds the threshold. Once the number of free RxBDs exceeds the threshold, the configuration for LFC may be safely modified.

Fix plan: Fixed in Rev 2.0

eTSEC58: 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

eTSEC59: False parity error at Tx startup

Description: The 10 KB TxFIFO comes out of reset in an unitialized state. Each FIFO entry is initialized as Tx frame data is written to it. Under certain internal resource contention conditions, the controller may read uninitialized data and falsely signal a parity error in IEVENT.

Impact: If parity errors are enabled before the first 10KB of Tx frame data is written to the TxFIFO, pause frames or Tx frames may trigger a false data parity error event.

Rev 1.0 only:

Also, the false parity error may cause FCS corruption on the transmitting frame, if there is one.

Workaround: Disable parity error detection by setting EDIS[DPEDIS]=1 until at least 10 KB of Tx data has been transmitted.

Fix plan: No plans to fix

eTSEC60: Tx data may be dropped at low system to Tx clock ratios

Description: The logic in the Ethernet controller has an asynchronous clock crossing between the Tx data FIFO (eTSEC system clock domain) and the MAC (Tx clock domain). Under some conditions between frames there is a short pulse from the Tx clock domain back to the eTSEC system clock domain which may be missed if the eTSEC system clock domain is less than 160% of the Tx clock domain. If the pulse is missed, the Tx drops 4–12 bytes of transmit data, without any error indication. There is no data or packet format dependency for this fail scenario.

Impact: Tx data may be dropped at low system to Tx clock ratios.

Workaround: Run eTSEC system clock at least 1.6x Tx clock.

Fix plan: Fixed in Rev 2.0

eTSEC61: Rx may hang if RxFIFO overflows

Description: If the memory subsystem is unable to keep up with incoming traffic, the Rx FIFO may fill up and overflow. If the RxFIFO fills up, the controller should gracefully drop packets. Instead, under certain conditions on the interface between the controller and the memory subsystem, the Rx will lock up and stop receiving without any error indication.

Impact: For low ratios from platform to Rx_clk and slow memory systems, the Rx FIFO may overflow and hang the Rx controller.

Workaround: To reduce the probability of an RxFIFO overflow, enable flow control by setting MACCFG1[Tx Flow] = 1.

Statistical lockup detection and recovery:

Lockup detection:

1. Enable debug mode in the controller by writing 0x00E00C00 to offset 0x000 (TSEC_ID1).
2. Periodically poll the state of the Ethernet controller by reading RPKT, RSTAT, and the register at offset 0xD1C. If RPKT has changed, the RSTAT[QHLTn] bits are clear, and the value of register offset 0xD1C has not changed, wait X*16 bit times, where X is the largest frame expected to be received on this interface, then read the value of register offset 0xD1C again. If it still has not changed, and RPKT has changed again, then the Rx controller may be locked up. If promiscuous mode is disabled (RCTRL[PROM] = 0), or if the controller is likely to receive and discard fragmentary packets (both of which may cause RPKT to increment for packets which are discarded before the RxFIFO) additional iterations may be required to reduce the probability of a false lockup detect.

There is no guaranteed algorithm to detect Rx lockup with zero false positives.

Lockup recovery:

1. Perform a graceful receive stop by setting DMACTL[GRS] = 1, and wait to ensure any outstanding prefetches are cleared. The wait time is system and memory dependent, but a reasonable worst-case time is the receive time for a 9.6 KB frame at 10/100/1000 Mbps). Note that if the Rx is truly locked up, IEVENT[GRSC] will never be set. The graceful receive stop also ensures that data and state are not corrupted during a soft reset if the lockup detection falsely detects a lockup due to rejected packets.
2. Toggle MACCFG1[Rx En] (set to 0, then set to 1).
3. Clear the graceful receive stop by setting DMACTL[GRS] = 0.

Fix plan: Fixed in Rev 2.0

eTSEC62: Rx packet padding limitations at low clock ratios

Description: There are two mechanisms that cause extra bytes to be inserted in front of the data in a received frame:

1. RCTRL[PAL] - packet alignment padding. A programmable mechanism for padding a frame with zeroes to achieve a particular alignment of data. Additionally if the 1588 time-stamping feature is enabled, the padding includes the 8 bytes of 1588 Rx timestamp data.
2. MACCFG2[PreamRxEn] – enables inserting the 8-byte preamble in front of the Rx frame data within the data buffer. These bytes are not accounted for in the value of RCTRL[PAL] setting.

At low clock ratios (less than 4:1 platform clock to MAC interface clock ratio), it is possible for the receive buffer to overflow when 24 or more extra bytes are inserted into the Rx data buffer. When this Rx buffer overflow occurs, the current Rx frame is dropped and the subsequent frame may be passed to memory without the expected padding bytes inserted.

The MAC interface clock rate is determined by the interface configuration and link data rate. Parallel MAC interfaces operating at 1000 Mbps, such as GMII and TBI, have a 125 MHz MAC interface clock rate. Parallel interfaces operating at either 100 Mbps or 10 Mbps have a MAC interface clock rate that is ¼ of the bit rate (25 MHz at 100 Mbps and 2.5 MHz at 10 Mbps). When operating in SGMII mode, the MAC interface clock rate is determined by the actual link data rate, exclusive of frame elongation bytes that are used when transferring 10 Mbps and 100 Mbps data across the link. In SGMII mode, the MAC interface clock rate is 125 MHz for 1000 Mbps data rate, 25 MHz for 100 Mbps data rate, and 2.5 MHz for 10 MHz data rate.

Impact: If the platform clock is less than 500 MHz and the eTSEC is operating at a 1000 Mbps data rate (regardless of interface configuration), the eTSEC cannot support inserting 24 or more total bytes (from padding, time-stamping and the preamble) in front of the Rx frame data.

Workaround:

- Limit total receive packet byte insertion via RCTRL[PAL], 1588 time-stamping, and Rx preamble enable to less than 24 bytes total when the platform clock is less than 500 MHz and the interface is operating at 1000 Mbps data rate.
- Limit the eTSEC data rate to 100 Mbps or less when the platform clock is less than 500 MHz. This can be accomplished by using an RGMII, MII, RMII, or SGMII interface in conjunction with setting MACCFG2[I/F Mode]=01.

Fix plan: Fixed in Rev 2.0

eTSEC63: False TCP/UDP checksum error for some values of pseudo header Source Address

Description: The Ethernet controller calculates the pseudo header checksum by first calculating the checksum for the individual fields of the pseudo header, then merging the checksums and carry bits. If the checksum for the Source Address (SA) field of the pseudo header is 0x1_0000 (16-bit checksum=0 with carry out=1), the carry bit is not included in the combined checksum, resulting in a false checksum error (RxFCB[ETU]=1). A pseudo header SA checksum of 0x1_0000 is only possible for IPv6frames, not IPv4.

Impact: False ETU indication when check sum for pseudo header SA is 0x1_0000 for IPv6 frames.

Workaround: If RxFCB[CTU]=1, RxFCB[ETU]=1 and RxFCB[IP6]=1, calculate the checksum for the SA field from the pseudo header. If this checksum equals 0x1_0000, then proceed to calculate the entire TCP checksum to be sure the checksum error is valid. If the SA checksum is not 0x1_0000, then the ETU is a valid checksum error indication.

Fix plan: Fixed in Rev 2.0

eTSEC64: 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

eTSEC65: 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: Partially fixed in Rev 2.0

Fixed for single-queue Tx bandwidth.

eTSEC66: Controller stops transmitting pause control frames

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

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

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

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

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

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

Options to minimize the frequency of the error condition:

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

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

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

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

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

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

Detection and recovery:

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

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

1. Enable BSY interrupts by setting IMASK[BSYEN] = 1 and EDIS[BSYDIS] = 0.
2. On receiving a BSY interrupt, write a 1 to IEVENT[BSY] to acknowledge the event
3. Assume the pause state machine is stuck and proceed with the following recovery mechanism:
    a. Set DMACTRL[GTS] = 1 to perform a graceful transmit stop
    b. Wait for IEVENT[GTSC] to be set, indicating stop complete
    c. Write a 1 to IEVENT[GTSC] to acknowledge the event
    d. Toggle MACCFG1[Tx_En] 1->0->1
    e. Set DMACTRL[GTS]=0 to clear the graceful transmit stop
    a. Set DMACTRL[GTS]=1 to perform a graceful transmit stop
    b. Wait for IEVENT[GTSC] to be set, indicating stop complete
    c. Write a 1 to IEVENT[GTSC] to acknowledge the event
    d. Clear MACCFG1[Tx_Flow]
    e. Wait 256 TX_CLK cycles
    f. Clear MACCFG1[Tx_EN]
    g. Wait 3 TX_CLK cycles
    h. Set MACCFG1[TX_EN] and MACCFG1[Tx_Flow]
    i. Set DMACTRL[GTS]=0 to clear the graceful transmit stop

Fix plan: Fixed in Rev 2.1

eTSEC67: ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

eTSEC68: 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. 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: Documentation update

The reference manual will be updated to require padding of all short Tx frames when in half-duplex mode (MACCFG2[Full Duplex] = 0).

eTSEC69: 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:

- Sequence a) FF_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

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

- Sequence b) FF_FF_FF_FF_FF_FF_01_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

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

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

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

- Sequence c) FF_FF_FF_FF_FF_FF_01_02_03_04_FF_FF_FF_FF_FF_FF_01...

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

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

Impact: The Ethernet controller 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

eTSEC70: 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)

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

Workaround: None

Fix plan: No plans to fix

eTSEC71: The value of TSEC_ID2 is incorrect

Description: The eTSEC1 and eTSEC2 TSEC_ID2 registers (0x24004, 0x25004) have a wrong default value of 0x00EC00F0, not according to the Reference Manual. The correct value, as in the Reference Manual, should be 0x00E000F0.

Impact: There is no functional impact.

Workaround: None.

Fix plan: No plans to fix

eTSEC72: Receive pause frame with PTV = 0 does not resume transmission

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

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

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

Fix plan: No plans to fix

eTSEC73: 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

eTSEC74: MAC: Rx frames of length MAXFRM or MAXFRM-1 are marked as truncated

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

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

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

Workaround: Option 1:

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

Option 2:

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

Fix plan: No plans to fix

eTSEC75: Misfiled Packets Due to Incorrect Rx Filer Set Mask Rollback

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

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

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

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

Rule #2: <Mask = 0x0000_8000>

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

Rule #3: <Mask = 0x0000_8000>

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

Rule #4: <Mask = 0x0000_8000>

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

Rule #5: <Mask = 0x0000_0210>

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

Rule #6: <Mask = 0x0000_8000>

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

Rule #7: <Mask = 0xFF00_FFFF>

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

Rule #8: <Mask = 0xFF00_FFFF>

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

Rule #9: <Mask = 0xFF00_FFFF>

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

Rule #10: <Mask = 0xFF00_FFFF>

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

Rule #11: <Mask = 0xFF00_FFFF>

etc.

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

Rule #8: <Mask = 0xFF00_FFFF>

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

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

Rule #9: <Mask = 0x0000_8000>

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

Rule #10: <Mask = 0x0000_8000>

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

Rule #11: <Mask = 0x0000_8000>

Mask should be 0xFF00_FFFF.

etc.

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

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

Workaround: Use one of the following workarounds:

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

Fix plan: No plans to fix

eTSEC76: Excess delays when transmitting TOE=1 large frames

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

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

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

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

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

Fix plan: No plans to fix

eTSEC 78: Controller may not be able to transmit pause frame during pause state

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

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

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

Workaround: None

Fix plan: No plans to fix

eTSEC79: 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: No plans to fix

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

Description: Devices: MPC8313E, MPC8313

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

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

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

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

Fix plan: No plans to fix

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

eTSEC-A004: User-defined preamble not supported at low clock ratios

Description: The Ethernet controller is not designed to inject user-defined preambles into Tx frames at eTSEC system clock to Tx clock ratios of less than 1.8:1. If the controller is run with a slower eTSEC system clock, the eTSEC may hang, underrun, or corrupt the transmitting frame.

Impact: User-defined Tx preamble may cause hang, underrun, or corrupted frames.

Workaround: Disable user-defined preamble Tx injection by setting MACCFG2[PreAmTxEn]=0.

Fix plan: Fixed in Rev 2.0

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 access to electrical idle detection circuitry in SGMII mode and there is no bit replacement for TBI SR[Link Status] to monitor.

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 reasons why RBYT may not increment (controller configuration errors, SerDes PLL issues, or MIB Counter RCDE incremented). Examination of system conditions may be necessary to determine why RBYT has not incremented.

Fix plan: No plans to fix

IEEE 1588_1: eTSEC IEEE 1588 reset bit

Description: There is a self clearing bit in the 1588 timer of eTSEC [tmr_soft_reset]. This bit is supposed to help the timer recover from any unpredicted fails in the timer without applying hard_reset, or when the timer clock is changed while the timer is active, or before enabling the timer it can/ should be given a soft reset to ensure a clean start. The first time the soft reset is used, it is cleared on its own, but any subsequent soft reset does not have any self-clearing function because of this erratum.

Impact: Clearing of this bit has to be done by software.

Workaround: Software needs to deassert it after ~20 cycles of asserting it.

Fix plan: Fixed in Rev 2.0

IEEE 1588_2: IEEE 1588 not supported in SGMII mode

Description: The eTSEC controller does not properly support the time-stamping of packets on the SGMII interface.

Impact: IEEE 1588 functionality is not supported in SGMII mode.

Workaround: None.

Fix plan: Fixed in Rev 2.0

IEEE 1588_3: FIPER outputs cannot be phase aligned to a specific point in time

Description: The IEEE Std 1588 logic is capable of generating up to three separate periodic output pulse signals. Although the period of each output pulse signal can be independently controlled by programming the corresponding FIPER registers, the output pulse signals cannot be phase aligned to a specific point in time.

Impact: The periodic output pulse signals cannot be phase aligned to a specific point in time. This is useful in many applications, including applications where the periodic output pulse signals must be asserted coincident with the exact seconds rollover point; this is used to demonstrate synchronization with a remote node.

Workaround: None for the FIPER output signals. However, software can generate a phase aligned PPS signal by programming one of the ALARM outputs to assert at the exact point of a seconds rollover. Immediately after the ALARM output is asserted, the software can reprogram the ALARM output to assert at the next seconds rollover point.

Fix plan: Fixed in Rev 2.0

IEEE 1588_4: ALARM output does not go active unless the ALARM comparison time is exactly equal to the current time value

Description: The ALARM outputs should be asserted when the current time meets or exceeds the comparison time value programmed in the corresponding ALARM registers. The erratum is that the ALARM outputs is only asserted when the current time is equal to the comparison time value programmed in the corresponding ALARM registers.

Impact: The ALARM outputs does not properly assert if the value of the current time exceeds, but does not equal, the comparison time value programmed in the ALARM registers. This can occur if the current time counter value is set to increment by the number of nanoseconds in the clock period, and the ALARM comparison time value happens to be set to a value between consecutive current time counter values.

Workaround: Set the current time counter increment period to be 1 (TMR_CTRL[TCLK_PERIOD] = 0x0001) instead of incrementing by the actual time period in nanoseconds. This setting ensures that the ALARM time comparison value exactly matches the current time counter value and properly assert the ALARM output signal.

When using the increment by period method (TMR_CTRL[TCLK_PERIOD] > 0x0001), ensure that the contents of the ALARM time comparison register are programmed such that the exact current time counter value is hit.

Fix plan: Fixed in Rev 2.0

IEEE 1588_5: 1588 external trigger timestamp does not include offset register value

Description: When a change of state on the external trigger input is detected, the eTSEC captures the time of this event in a timestamp register. The time that is recorded in the timestamp register corresponds to the current system time, which is composed of the sum of the 64-bit time counter and the offset register. The erratum is that the external trigger input timestamp register captures only the value of the 64-bit time counter instead of the current system time. The value of the offset register is not included in the captured timestamp. Note that, additionally, changing offset values on-the-fly results in an incorrect timestamp value.

Impact: Because the value of the offset register is not included in the captured external trigger timestamp, the captured value does not reflect the current system time of the local node.

Workaround: When an external trigger event causes a timestamp to be captured, software must calculate the true current system time of the event by adding the current value of the offset register to the captured timestamp value. The offset value must not be modified on the fly.

Fix plan: Fixed in Rev 2.0

IEEE 1588_6: 1588 ALARM output comparison value does not include offset register value

Description: The ALARM outputs should be asserted when the current system time meets or exceeds the comparison time value programmed in the corresponding ALARM registers. The time that is compared against the values in the ALARM register is the current system time, which is composed of the sum of the 64-bit time counter and the offset register. The erratum is that the values in the ALARM registers are compared against only the value of the 64-bit time counter instead of the current system time. The value of the offset register is not included in the ALARM register comparison.

Impact: Because the value of the offset register is not included in the ALARM time comparison, ALARM output signals may not be asserted as expected at the appropriate current system time.

Workaround: Software must subtract the offset register value from the current system time corresponding to when the ALARM output signal should be asserted, and use this value to program the ALARM comparison register value.

Fix plan: Fixed in Rev 2.0

IEEE 1588_7: ALARM output signal deasserts prematurely during current system time rollover condition

Description: The ALARM outputs should be asserted when the current system time meets or exceeds the comparison time value programmed in the corresponding ALARM registers, and they should remain asserted until the corresponding ALARM register upper and lower registers are reprogrammed. When the ALARM lower register is programmed, the corresponding ALARM output signal is disarmed; the ALARM output signal is re-armed when the ALARM upper register is programmed. The erratum is that if the ALARM output signal is asserted, it prematurely deasserts without being disarmed when the current system time rolls over to all zeros.

Impact: The ALARM output signals do not remain asserted during a current system time rollover condition.

Workaround: None.

However, the effects can be mitigated by not setting the ALARM comparison value more than (MAX (alarm value) – TMR_CNT [TCLK_PERIOD]) and ensuring that circuits connected to the ALARM outputs is not affected by deassertion during rollover.

Fix plan: Fixed in Rev 2.0

IEEE 1588_8: Writing Offset registers during use may yield unpredictable results

Description: The current system time, which is used to timestamp packets and events, is composed of the sum of the 64-bit time counter and the offset register. The offset register can be updated (written) by the CPU. The erratum is that CPU writes to the OFFSET register may result in temporarily unstable values on the register outputs, which can result in an incorrect calculation of the current system time, as well as incorrect timestamps.

Impact: Applications that require the CPU to periodically update the OFFSET register while it is being actively used may not work correctly due to unpredictable current time values.

Workaround: None.

Although the OFFSET register can be updated during initialization, care must be taken to ensure that it is not updated during active use. If the register is updated, ensure that any timestamps recorded during that time are ignored and not used as they may be in error.

Fix plan: Fixed in Rev 2.0

IEEE 1588_9: 1588 reference clock limited to 1/2 controller core clock in asynchronous mode

Description: If the eTSEC system clock is not at least twice the 1588 reference clock frequency, the PTP ID from TxFCB[VLCTL] may not be captured correctly in the TMR_TXTS1/2_ID (transmit time stamp identification) register.

Impact: A fast 1588 reference clock may lead to an invalid PTP ID in the TMR_TXTS1/2_ID register.

Workaround: Run the 1588 reference clock synchronously (TMR_CTRL[CKSEL] = 01). Refer to IEEE 1588_15.

Fix plan: Fixed in Rev 2.0

IEEE 1588_10: eTSEC IEEE Std 1588 logic only requires one offset register per counter instance

Description: The IEEE Std. 1588 logic generates a current system time that can be used to timestamp packets that are transmitted and received by each eTSEC. The current system time is composed of the sum of the 64-bit time counter and the offset register. All Ethernet ports within a device need to use the same current system time reference in order for the timestamping operations across all ports to have a consistent meaning to higher layer application software. Note, however, that this does not preclude the application software from maintaining different offsets per port in software. The erratum is that if an independent OFFSET register is used per port, then the Ethernet ports could have different current system times if the OFFSET registers are programmed differently.

Impact: If the OFFSET registers (TMR_OFF_H/L) for different eTSEC ports have different values, the timestamp values for the ports can have different current time references.

Workaround: Software must ensure that all IEEE Std 1588 OFFSET registers are programmed to the same value to ensure that all eTSEC ports use the same time reference.

Fix plan: Fixed in Rev 2.0

IEEE 1588_11: 1588 reference clock pulse required between writes to TMR_ALARMn_L and TMR_ALARMn_H

Description: The 1588 alarm event is enabled by first writing the TMR_ALARMn_L register and then writing the TMR_ALARM_H register. If there isn't at least one 1588 reference clock pulse between the two writes, the alarm is not enabled, and the corresponding alarm event does not occur even when the current time reaches and passes the alarm value. The 1588 reference clock is selected by TMR_CTRL[CKSEL].

Impact: If writes to the two ALARM registers occur faster than a 1588 reference clock period, the alarm event may never fire. This happens when TSEC_1588_CLK is the selected 1588 reference clock, which is slower than the eTSEC system clock.

Workaround:
- Option 1: Add a delay of at least one 1588 reference clock period between the write to TMR_ALARMn_L and the write to TMR_ALARMn_H, slowing down the writes.
- Option 2: Follow a write to TMR_ALARMn_L with a read to it, and then a write to TMR_ALARMn_H.

Fix plan: Fixed in Rev 2.0

IEEE 1588_12: 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: Partially fixed in Rev 2.0

Timer alarm 1- Fixed in Rev 2.0.

Timer alarm 2 - No plans to fix

IEEE 1588_14: TxPAL timestamp uses TxBD snoop enable instead of Tx data

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

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

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

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

Fix plan: No plans to fix

IEEE 1588_15: Use of asynchronous 1588 reference clock may cause errors

Description: There is 1588 clock-domain information related to the timestamp that must cross to the eTSEC system clock domain and is not properly synchronized. During 1588 negotiation, S/W must also write some of the 1588 registers. The new register values then cross from the eTSEC system clock domain to the 1588 clock domain where the 1588 state machines reside. That crossing is also unsynchronized, with the effect that the register values is unstable for a few cycles and may cause improper operation of the state machines.

Impact: If the 1588 clock and the eTSEC system clock are asynchronous, then there is no guarantee that all bits in the 1588 registers and state machines are stable when they are latched by the eTSEC system clock. This can lead to cases where counter values 'appear' to jump forward or run backwards (due to the settling of the bits), or where state machines transition improperly to new states.

Workaround: This asynchronous boundary problem can be avoided by clocking the 1588 clock domain using the eTSEC system clock, thus guaranteeing that the two domains are synchronous with each other. The eTSEC system clock is selected as the 1588 reference clock by setting TMR_CTRL[CKSEL] = 01.

Fix plan: Fixed in Rev 2.0

This is fixed on MPC8313E revision 2.0 silicon, which supports the following IEEE Std 1588 clock source options:

- eTSEC internal system clock
- External clock applied to the TSEC_1588_CLK input pin, for example, oscillator, VCXO, GPS, etc.
- RTC clock

IEEE 1588_16: Odd prescale values not supported

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

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

Impact: Odd 1588 timer prescale values are not supported.

Workaround: Use only even timer prescale values.

Fix plan: No plans to fix

IEEE 1588_17: Cannot use inverted 1588 reference clock when selecting eTSEC system clock as source

Description: The source of the 1588 reference clock can be selected with TMR_CTRL[CKSEL] register bit field, while the TMR_CTRL[CIPH] register bit field allows the user to invert the selected 1588 reference clock.

If the eTSEC system clock is chosen as the source for the 1588 reference clock (TMR_CTRL[CKSEL]=01), then selecting an inverted version of such clock (by setting TMR_CTRL[CIPH]=1) results in an improperly controlled 1588 reference clock, and boundedly undefined errors.

Impact: Cannot select an inverted 1588 reference clock if the eTSEC system clock is chosen as the source for the 1588 reference clock.

Workaround: None

Fix plan: No plans to fix

IEEE 1588_18: FIPER's periodic pulse phase not realigned when the 1588 current time is adjusted

Description: After the FIPER is activated and phase aligned to a specific point in time, it is out of phase to the current time if the current time is subsequently adjusted via updates to the TMR_CNT_H/L or the TMR_OFF_H/L register. The FIPER continues to pulse at the set periodic intervals, ignoring the current time update.

Impact: The FIPER outputs need to be stopped, re-initialized, and restarted if current time adjustments are made through the TMR_CNT_H/L or the TMR_OFF_H/L register.

Workaround: To stop and restart FIPER pulsing in phase to the updated current time if TMR_CTRL[FS]=0:

1. Write any value to TMR_ALARM1_L to disable the ALARM
2. Disable the FIPER pulse, and program it to be re-enabled by ALARM by setting TMR_CTRL[FIPER Start]=1
3. Update the 1588 current time as desired
4. Set TMR_ALARM1_L to the desired value to align the next FIPER pulse
5. Set TMR_ALARM1_H to the desired value to align the next FIPER pulse and enable the alarm.
6. After ALARM1 event, TMR_TEVENT[ALM1], clear TMR_CTRL[FIPER Start]

To stop and restart FIPER pulsing in phase to the updated current time if TMR_CTRL[FS]=1:

1. Write any value to TMR_ALARM1_L to disable the ALARM and FIPER pulse
2. Update the 1588 current time as desired
3. Set TMR_ALARM1_L to the desired value to align the next FIPER pulse
4. Set TMR_ALARM1_H to the desired value to align the next FIPER pulse and enable the alarm.

Fix plan: Fixed in Rev 2.0

IEEE 1588_19: Tx FIFO data parity error (DPE) may corrupt Tx timestamps if TMR_CTRL[TRTPE]=1

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

Example:

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

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

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

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

OR

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

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

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

Fix plan: No plans to fix

IEEE 1588_20: eTSEC 1588: Write to reserved 1588 register space causes system hang

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

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

Reads to 1588 shared registers from any other eTSEC return 0's. Writes to 1588 shared registers from any other eTSEC should be silently dropped, but instead cause a hang of the memory-mapped register IP interface.

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

Workaround: Do not write to 1588 shared registers from any eTSEC other than eTSEC1.

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: Plan to fix in Rev 2.2

eLBC1: LTESR[CS] error issue

Description: LSOR is used as a special operation by reporting chip select errors in LTESR[CS]. This special operation initiates a correct response as expected by the OP field. For example, when OP = 01, that is, the UPM RAM Array is to be written and LSOR is written, then correct data from MDR is written into the UPM RAM word, but this gives a chip select error. This works fine when JTAG is used as the master but gives a CS error when the core is the master. As an alternative, if a dummy write is used instead of a write to LSOR, it does not report a LTESR[CS] error.

Impact: A write to LSOR performs the correct intended operation, but this causes an LTESR[CS] error. So, software will report error if it is monitoring the LTESR register.

Workaround: A dummy transaction can be used as an alternative to writing to LSOR, which performs the same operation.

Fix plan: Fixed in Rev 2.0

eLBC2: UPM does not have indication of completion of a Run Pattern special operation

Description: A UPM or FCM special operation is initiated by writing to MxMR[OP] or FCM[OP] and then triggering the special operation either through the LSOR register or by performing a dummy access to the bank.

The UPM and FCM are expected to have different indications of when the special operation is completed. The FCM will see the LTESR[CC] bit set when such a special operation is completed. The UPM will see MxMR[MAD] increment when a write to or read from UPM array special operation completes. However, the UPM does not have any status indication of completion of the run pattern special operation.

The following two scenarios could be affected by this erratum:

1. A UPM Run Pattern special operation is initiated and a second UPM command is issued before the Run Pattern is completed.

   If the above scenario occurs the programmed mode registers could be altered according to the second operation and cause the current/first operation to encounter errors due to mode changes in the middle of the operation. Note that if the second command issued is a Run Pattern operation and it does not change the mode registers, the first operation should not encounter errors.

2. A write to LSOR initiates a UPM Run Pattern special operation and then LSOR is written too again, to initiate a second special operation that requires the mode registers to change while the first Run Pattern operation is in progress.

   If the second special operation issued does not change the programming mode of the first Run Pattern operation because the operation is either exactly the same as the first or it requires programming of an independent set of registers then the Run Pattern operation should not encounter errors.

The behavior of the eLBC is unpredictable if the Run Pattern special operation mode is altered between initiation of the operation and the relevant memory controller completing the operation.

Impact: Because of this erratum, when a UPM run pattern special operation is to be followed by any other UPM command for which the mode registers need to change, the Run Pattern operation may not be handled properly. Software does not have any means to confirm when the current run pattern special operation has completed so that register programming for the next operation can be done safely.

Workaround: None

Fix plan: Fixed in Rev 2.0

eLBC3: eLBC NAND Flash memory has an ECC syndrome that collides with the JFFS2 marker in Linux

Description: The current NAND Flash memory ECC location collides with the JFFS2 marker in Linux. There is no industry specification about where the ECC syndrome should be placed.

Impact: Cannot boot from NAND flash if flash includes Linux JFFS2 markers.

Workaround: Disable hardware ECC and use software ECC instead. But hardware ECC is enabled by default during 4K boot phase and can only be disabled once boot is over for remaining data transfers. If the software can handle this, by using hardware ECC for the 4K boot block and software ECC for the rest of the NAND Flash, this workaround would work well.

Fix plan: Fixed in Rev 2.0

eLBC5: LTEATR and LTEAR may show incorrect values under certain scenarios

Description: The eLBC IP acks any transaction request when FCM special operation is in progress. In such a scenario when any one of the errors/events occur (such as Bus Monitor Timeout, Write Protect, Parity error, Atomic error, FCM command completion, or UPM command completion except for the CS error), the registers LTEAR and LTEATR capture the address and attributes of the most recently req-acked transaction instead of the FCM special operation that caused error. Hence indeterministic value may show up in those registers.

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

Workaround: None

Fix plan: No plans to fix

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

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

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

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

Fix plan: No plans to fix

eLBC-A002: Core may hang while booting from NAND using FCM

Description: When FCM is selected as the boot ROM controller via power-on-reset configuration, eLBC automatically loads 4K Bytes page of boot code into the FCM buffer RAM. These 4K Bytes consist of 8 pages (512 bytes each) of small-page NAND flash OR 2 pages (2048 bytes each) of Large-page NAND flash. The core begins execution as soon as the first page is loaded. If the core tries to execute an instruction that has not been loaded yet, unpredictable behavior may result.

Impact: Core might not be able to boot from NAND.

Workaround: Before execution of any instruction located beyond the first page, wait for LTESR[CC] (offset 0xB0) bit to be set. The bit indicates that all the pages have been loaded from NAND to 4K Bytes FCM buffer RAM.

Fix plan: No plans to fix

PCI2: NXT_PTR field of hot swap register in PCI power management

Description: PCI has a set of registers in the form of a link list showing PCI capability in both host and agent mode. The first capability register shows that PCI has a hot swap capability. Software finds the capability pointer from this register. In host mode, the NXT_PTR field of the hot swap register shows 0x80 describing power management capability, but in AGENT mode NXT_PTR shows 0x00, which means software does not know that power management capability is available in agent mode.

Impact: When software reads the PCI capability list registers, it finds that for the device agent, hot swap is the first and last capability (because in the hot swap capability register NXT_PTR is showing 0x00). Therefore, the host software deduces that the device is not capable of handling any PCI power management features.

Workaround: In hardware validation, power management capability can still be accessed by directly configuring PMCCR registers.

Fix plan: Fixed in Rev 2.0

PCI15: Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang

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

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

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

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

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

Workaround: Hardware workaround:

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

Software workarounds:

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

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

Fix plan: No plans to fix

PCI19: Dual-address cycle inbound write accesses can cause data corruption

Description: When using a dual-address cycle (DAC) for inbound write accesses and when the IOS is full, (that is, a previous PCI request to the IOS is being retried), the PCI overwrites the address for the IOS with the new address from the bus, despite the transaction being retried on the PCI bus.

Impact: Dual address cycle (DAC) feature cannot be sustained by the device while working as PCI target. As PCI initiator, DAC is not supported.

Workaround:
- When operating in host mode, map inbound windows to 32-bit addressing (below 4G addressing space) where this type of application is allowed.
- When operating in agent mode, the above workaround is not valid. The host is mapping the agents according to the address type in configuration registers, which, in this case, is '10'. Thus, it could map anywhere in the 64-bit address space.

Fix plan: No plans to fix

PCI20: PCI controller may hang when returning from "PCI pins low" state

Description: When PCI_GCR[PPL] is set, the output and bidirectional PCI bus signals are forced low to allow other PCI bus clients to switch off their power supply, putting the PCI controller into the “PCI pins low” state. When returning to an operational state (clearing the PCI_GCR[PPL] bit), the PCI controller may remain in a PCI “non-idle” state and not be able to perform any further PCI transactions.

The sequence of the failure is as follows:

1. While the PCI bus is inactive, all external bus requests to the PCI arbiter are blocked (by setting PCI_GCR[BBR]), and PCI pins are pulled-low (by setting PCI_GCR[PPL]) to enter “PCI pins low” state.
2. In “PCI pins low” state, the PCI controller state-machine still remains active. It is actively tracking the bus signals and is expecting them to meet the AC timing specifications.
3. When PCI_GCR[PPL] = 0, the PCI control signals start to rise to “1” logic, pulled by the bus pull-up resistors only.
4. The AC timing specifications of the rising signals at this moment may not always be met, causing a timing violation to occur and, in turn, causing the PCI controller to hang.

Impact: The PCI controller may hang when attempting to return from the “PCI pins low” state.

Workaround: Use one of the following options:

- Before Setting PCI_GCR[BBR] and PCI_GCR[PPL] to enter “PCI pins low” state, Clear PCI command bit1 to disable PCI Memory Space. While coming back from “PCI pins low” state, after clearing PCI_GCR[PPL] = 0, read back the PCI_GCR[PPL] to ensure the operation was completed. Immediately afterward, turn off the clocks to the PCI controller (clearing SCCR[PCICM]) to ensure that the PCI controller will stop during the slow rise time of the PCI signals. After waiting up to 10 microseconds, turn the PCI clocks ON and resume the operations to enable the full functionality of PCI bus. Resume the sequence to enable the PCI bus to return to full functionality. At last, Set PCI Command bit1 to enable PCI memory space.
- Connect a wire between an unused GPIOn[x] pin and the FRAME signal. Make sure that this GPIOn[x] is an input while HRESET is active and after its negation (GPnDIR[x] = 0). During the resume procedure, before clearing PCI_GCR[PPL], set the GPIOn[x] data register to '0' (GPnDAT[x] = 0), and then set GPIOn[x] to be an output signal (GPnDIR[x] = 1). Only now, clear PCI_GCR[PPL]. Wait up to 10 microseconds, allowing the PCI signals (except FRAME) to return to a steady state. Set the GPIOn[x] to '1' (GPnDAT[x] = 1), then set GPIOn[x] as an input (GPnDIR[x] = 0). Resume the sequence to enable the PCI bus to return to full functionality.

Fix plan: No plans to fix

PCI22: PCI input and output hold from clock (t_{PCIXKH} and t_{PCKHOX}) do not meet specification

Description: The PCI controller may not meet the minimum specifications for input hold from clock (t_{PCIXKH}, 0 ns) at 33 and 66 MHz and output hold from clock (t_{PCKHOX}, 1 ns) at 66 MHz.

Impact: May lead to reduced input and output hold times.

Workaround: Use a hold time of 0.2 ns at both 33 and 66 MHz. Use a hold time of 0.8 ns at 66 MHz.

To meet the output hold specification ($t_{PCIKHOX}$) at 66-MHz PCI operation, write 0x2AAA_0002 to address (IMMR_BASE+ 0x0000_0124) before using any PCI pad. Output hold specifications are met at 33-MHz PCI operation and no work around is needed.

Fix plan: Fixed in Rev 2.1

Fixed t_{PCIXKH} in Rev 2.1 silicon and t_{PCKHOX} in Rev 2.0.

PCI23: Device locks up if a wake-up event occurs immediately after D3-warm instruction by e300

Description: When the e300 core asks the PMC to transition the device into D3-warm state (power saving mode in which some blocks are powered down), there exists a window immediately after this during which if any wake-up event occurs, the device locks up and needs a reboot thereafter. Ideally, the transition to D3-warm power saving mode is supposed to continue even in the case of a wake-up event immediately after. If the wake-up event occurs after this window the transition completes as expected.

Impact: The device locks up if a wake-up event occurs immediately after D3-warm instruction by e300.

Workaround: None.

Fix plan: Fixed in Rev 2.0

PCI24: While ICACHE on, critical data word access is corrupted

Description: When the PCI is mastering a burst with a non-aligned address (critical word first), it does not use the cacheline wrap addressing mode of the PCI bus. Rather, it uses the linear addressing mode and performs the burst in two parts: until the cacheline boundary and then from the beginning of the cacheline. The problem is that it wraps the address to a 64-byte boundary instead of a 32-byte boundary.

Impact: Instruction fetch data may be corrupted if ICACHE is enabled and fetched from the PCI space.

Workaround: While fetching the core instructions from PCI space, do not the enable the ICACHE.

Fix plan: Fixed in Rev 2.0

DMA2: Data corruption by DMA when destination address hold (DAHE) bit is used

Description: There can be corruption of the DMA data under the following conditions:

- DMAMR[DAHE] = 1 (destination address hold)
- DMAMR[DAHTS] = 10 (4 bytes) or 11 (8 bytes)
- DMA source address is not aligned to the transaction size specified by DAHTS
- The source port width is smaller than the destination transaction size or the source port returns valid read data only in the valid byte lanes

Examples of error condition are as follows:

- DAHTS is 8 bytes and the source port is a 32-bit PCI bus
- The source memory space is on the PCI bus and is not prefetchable

Impact: Corrupted data written to the destination peripheral or memory.

Workaround: Use one of the following options:

- Use a source address aligned to the destination transaction size
- Do not access any DMA registers while this type of DMA transfer is active

Fix plan: No plans to fix

IPIC2: eTSEC interrupts are swapped in the IPIC

Description: The eTSEC interrupt vector values were incorrectly switched as follows:

32 = TSEC2 ERR

33 = TSEC2 RX

34 = TSEC2 TX

35 = TSEC1 ERR

36 = TSEC1 RX

37 = TSEC1 TX

The correct eTSEC interrupt vector values are as follows (refer to the MPC8313ERM):

32 = TSEC1 TX

33 = TSEC1 RX

34 = TSEC1 ERR

35 = TSEC2 TX

36 = TSEC2 RX

37 = TSEC2 ERR

They bit fields for the eTSEC interrupts in the SIPNR_H / SIFCR_H / SIMSR_H registers were incorrectly switched as follows:

0 = TSEC2 ERR

1 = TSEC2 RX

2 = TSEC2 TX

3 = TSEC1 ERR

4 = TSEC1 RX

5 = TSEC1 TX

The correct bit field assignments are as follows (refer to the MPC8313ERM):

5 = TSEC2 ERR

4= TSEC2 RX

3 = TSEC2 TX

2 = TSEC1 ERR

1 = TSEC1 RX

0 = TSEC1 TX

Impact: If legacy software is used, there may be a problem getting eTSEC interrupts correctly.

Workaround: Change the value of the eTSEC interrupts vector in the software accordingly to match the silicon behavior.

Fix plan: Fixed in Rev 2.0

USB1: USB jitter issue

Description: In high speed mode, the USB PHY transmitter eye diagram is not meeting the specifications. USB 2.0 specifications ask for a particular mask to be met in the eye diagram. The USB PHY transmitter eye diagram on this device is out of compliance with this mask on the time axis by around 70 ps. This is causing the test to fail. High jitter can be the cause in the USB PLL.

Impact: The USB PHY, is not USB 2.0 compliant. This bug leads to higher bit error rates when working with other PHY(s) and fail in compliance testing.

This bit error causes packet retransmission to happen and thus loss of bandwidth.

Workaround: None.

Fix plan: Fixed in Rev 2.0

USB15: 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

USB18: USB shows device attached even when the device has been detached in full-speed mode

Description: When the device is in device mode; we have to enable the run/stop bit (RS) in the USB command register (USBCMD) to enable a pull-up on the D+ line to initiate a attach event. But when we detach the device either with software or by disconnecting the cable or giving a power-on reset/hard reset, even then the D+ line shows a pull-up.

This means the device is still attached (pull-up on D+ is still there) which causes the device to give an attach event to the Host even though RS is zero. The pull-up gets disconnected only when we power-off the device.

Impact: The device gives attach to the host although it is disconnected by resetting RS (through power-on reset, hard reset) bit and may not be attached to another host.

Workaround: None.

Fix plan: Fixed in Rev 2.0

USB19: USBDR in Host mode does not generate an interrupt upon detection of a CRC16/PID/Timeout error when a received IN data packet is corrupted

Description: USB dual role controller (USBDR) in Host mode does not generate an interrupt and does not set the respective bit in the USBSTS register upon detection of a CRC16/PID/Timeout error when a received IN data packet is corrupted. The error information, however, is written into the status field of the corresponding data structure and is not lost.

Impact: None

Workaround: The software driver should always check the status field of the transfer descriptor.

Fix plan: No plans to fix

USB20: Problem with configuration of RCWH/CFG_RESET_SOURCE when ULPI intended

Description: USB is muxed with eTSEC1. The default mode of operation depends on the eTSEC1 mode selected using RCWH.

eTSEC functionality is selected as the default mode for eTSEC1 pins when RTBI mode is selected. For all other eTSEC modes, USB functionality is selected for multiplexed pins.

The USBDR_STP pin is muxed with the TSEC1_TXD0 pin. For correct operation of the external ULPI PHY, we need the STP pin to be asserted until the ULPI PHY is enabled by the USB controller.

As long as RTBI mode is not selected, there are no issues.

Impact: When RTBI is selected using RCWH, eTSEC1 functionality is selected for multiplexed pins and the eTSEC controller drives TSEC1_TXD0 as low (as 0), which results in malfunctioning of the ULPI PHY.

Workaround: Customers running the USB in ULPI mode should not select RTBI for eTSEC1, neither with RCWH nor with CFG_RESET_SOURCE=4'b1000.

Fix plan: No plans to fix

USB21: 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: None

Workaround: None

Fix plan: Fixed in Rev 2.0

USB25: 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

USB26: NackCnt field is not decremented when received NYET during FS/LS Bulk/ Interrupt mode

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

USB27: When an ACK or NAK is sent from the device in response to a PING, the CERR counter value is not being reset to the initial value

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

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

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

Fix plan: No plans to fix

USB28: In device mode, when receiving a Token OUT, if a Rx flush command is issued at the same time, to the same endpoint, the packet will be lost

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

USB29: Priming ISO over SOF will cause transmitting bad packet with correct CRC

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

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

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

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

Workaround: There is no workaround.

Fix plan: No plans to fix

USB30: High speed output impedance fails marginally.

Description: The USB high-speed standard requires the impedance of each output including the contribution of a series resistor (RS) to be 45 Ω ±10%. MPC8315 fails this spec marginally, with measured value as 45 Ω –12%, at temperatures of below 0°C.

Impact: This marginal impedance failure of the standard has not shown to have any major impact in practical applications. In addition, the same impedance test passes in the peripheral compliance. The peripheral compliance requires the contribution of RS to be 45W±15%.

Workaround: None

Fix plan: No plans to fix

USB31: Transmit data loss based on bus latency

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

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

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

Fix plan: No plans to fix

USB32: Missing SOFs and false babble error due to Rx FIFO overflow

Description: Devices: MPC8313E, MPC8313

When in Host mode, if an Rx FIFO overflow happens close to the next Start-of-Frame (SOF) token and the system bus (CSB) 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

USB33: No error interrupt and no status will be generated due to ISO mult3 fulfillment error

Description: Devices: MPC8313E, MPC8313

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

USB34: 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

USB35: Core device fails when it receives two OUT transactions in a short time

Description: Devices: MPC8313E, MPC8313

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

USB36: CRC not inverted when host under-runs on OUT transactions

Description: Devices: MPC8313E, MPC8313

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

USB37: OTG Controller as Host does not support Data-line Pulsing Session Request Protocol

Description: Devices: MPC8313E, MPC8313

An OTG core as a Host must be able to support at least one Session Request Protocol (SRP) method (VBUS or Data-line Pulsing), but OTG as Device must support and use both when attempting SRP.

As our OTG controller as a Host fully supports VBUS pulsing, the SRP will always be successful and the impact of this issue is minor. However, the recent OTG 2.0 specification removes the VBUS pulsing SRP method, making the Data-line Pulsing a mandatory SRP detection for host controllers.

Impact: When the OTG core is acting as a Host, and VBUS is turned off, and the attached Device attempts to perform a Session Request Protocol (SRP) by using Data-line Pulsing, it will not be recognized by the Host.

Also, when doing role switching (HNP) and becoming a Host, a SE0 is forced in the line causing the OPT TD5.4 test to fail.

Workaround: The termsel will be changed from '0' to '1' when in reset and in the state after reset (PORT_DISABLE). As this signal is the same if the controller is host or device, the termsel was changed in both cases.

Software workaround is possible for the HNP situation only. With this workaround, it is possible to pass the OPT TD5.4 test. The software must assert core mode to device (USBMODE.CM) and Run/Stop bit (USBCMD.RS) to '1' just after the controller ends reset (wait until USBCMD.RST is '0' after setting it to '1').

This issue does not prevent OTG 1.3 SRP, since it is possible to do VBUS pulsing. This correction extends to OTG 2.0 support, since Data-line Pulsing is mandatory.

Fix plan: No plans to fix

A-003817: USB Controller locks after Test mode "Test_K" is completed

Affects: USB

Description: Devices: MPC8313E, MPC8313

Previously known as USB38

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

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

Workaround: None

Fix plan: No plans to fix

USB-A001: Last read of the current dTD done after USB interrupt

Description: Devices: MPC8313E, MPC8313

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.

Impact: Two different situations may occur:

- The case of a single dTD with the Interrupt on Completion (IOC) bit set. In this case, if the interrupt handling routine has a lower latency than the bus arbitration of the dTD read after the interrupt is posted (at this time the software will find the Active bit cleared - dTD retired by hardware), the software may clear or re-allocate the data structure memory location to other applications, before the last read is performed.
- The case of multiple dTDs with the IOC bit set. In this case, if the latency handling the interrupt is too long, the following might occur:
  a. The core could assert the interrupt when it completes dTD1.
  b. Proceed to execute the transfer for dTD2.
  c. By the time the core 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.
  d. The core then re-reads dTD2 and finds corrupted data. If the T bit is zero or the Active field in non active then the core will not re-prime.

Workaround: None

Fix plan: No plans to fix

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

Description: Devices: MPC8313E, MPC8313

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-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-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

SPI5: Selection of GPIO functionality on $\overline{\text{SPISEL}}$ signal causes MME in SPI

Description: If SPI is being used as the master and SICRL[SPI_D] is set to 11, a multiple master error (MME) occurs.

Impact: While using SPI in a master mode, GPIO31 cannot be used.

Workaround: SICRL[SPI_D] shall be set to 00 whenever SPI is used in the master mode. As described in the reference manual, if the MPC8313E SPI is used in a single-master mode, an external pull up on SPISEL is required.

Fix plan: Fixed in Rev 2.0

General10: Incorrect isolation for mux control for LCLK0/LCLK1 in low power mode

Description: When the device transitions to the low power state, D3 warm, a high frequency clock starts outputting on pads LCLK0 and LCLK1. The frequency of the clock on LCLK0 is equal to the CSB frequency and the frequency on LCLK1 is twice that of the CSB frequency.

Impact: If the connected memory device uses LCLK0 or LCLK1, check that the device does not malfunction on this frequency. Please note that chip selects are inactive at that time.

Workaround: Use an asynchronous memory.

Fix plan: Fixed in Rev 2.0

General11: DUART input high voltage does not meet the specification

Description: The DUART block may not meet the minimum specification for the input high voltage, V_{IH}.

Impact: The V_{IH} voltage is marginal at 2.0V and may not be interpreted as logic High.

Workaround: Ensure that V_{IH} minimum is 2.1V.

Fix plan: No plans to fix

General12: CSB deadlock

Description: Case 1: Instruction Caching is disabled for PCI space

If the e300 core initiates an instruction read from the PCI outbound space and, before its completion, starts another high priority data read from the same cache-line, a deadlock on the CSB results.

The gasket between the CSB and I/O sequencer (IOS) will ARTRY the first instruction read due to a 'delayed read' from PCI space. The gasket does not accept any further transactions within the same cache line until the previous one completes. When the e300 core performs a high priority data read after initiating an instruction fetch, it does not rerun the instruction read before the data load has been serviced. On the other hand, CSB2IOS does not serve data read from the same cache line until the instruction read completes. Thus, both the e300 core and CSB2IOS respectively wait for the data read and instruction read to complete, hence resulting in a deadlock.

Case 2: Instruction Caching is enabled for PCI space

When Instruction Caching is enabled for PCI space, e300 fetches complete cache-line from instruction space before it starts to execute it. Now if instruction in currently executed cache-line loads data from next cache-line, then it will create same scenario as Case 1 above.

Here core would initiate instruction fetch from next cache-line. On encountering data-load instruction (before instruction fetch completion), it will initiate high priority data read from next cache-line. As a result, the gasket between the CSB and I/O sequencer (IOS) will again have instruction read and data read from same cache-line with high priority data read following instruction read. Hence it will result in deadlock on CSB bus.

Impact: Deadlock on CSB. The gasket between the CSB and IOS checks the 32-bit address, transfer-type, transfer size, and transfer burst fields before deciding on a further course of action. If all four fields are identical to some previous buffered transaction, the gasket will generate ARTRY until read data is available. If any of the fields are different, the gasket will treat that as a different transaction and forward it to PCI.

Workaround: Use one of the following workarounds:

- Instruction space and data space should be kept mutually exclusive of each other.
- When executing a code from the PCI outbound space, data reads should not fall on the same cache line as an ongoing instruction read.

Fix plan: Fixed in Rev 2.0

General15: Electrostatic discharge (ESD) may fail to meet the 500 V charged device model (CDM)

Description: CDM (charged device model) ESD testing has shown that some pins, in particular the high speed SGMII PHY pins do not meet the 500 V CDM ESD criteria. CDM passes at 400 V.

The following pins are impacted:

TXA, TXA, TXB, TXB, SD_IMP_CAL_TX, SDAVDD, XPADVDD, XCOREVDD, SD_PLL_TPA_ANA

Impact: Excessive ESD can cause damage to the device.

Workaround: Ensure equipment used in handling of the device is properly grounded. Ensure parts are handled in an environment that is compliant with the current ESD standard.

Fix plan: No plans to fix

General16: 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

General17: DUART: Break detection triggered multiple times for a single break assertion

Description: A DUART 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 ULSRn and checking for BI=1. This read to ULSRn will clear the BI bit. Once the break is detected, the normal handling of the break condition is to read the URBR to clear the ULSRn[DR] bit. The expected behavior is that the ULSRn[BI] and ULSRn[DR] bits will not get set again for the duration of the break signal assertion. However, the ULSRn[BI] and ULSRn[DR] bits will continue to get set each character period after they are cleared. This will continue 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 ULSRn[DR] being set.

Impact: The ULSRn[BI] and ULSRn[DR] bits will 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 ULSRn is read and ULSRn[BI]=1. To prevent the problem from occuring, perform the following sequence when a break is detected:

1. Read URBRn, which will return a value of zero, and will clear the ULSRn[DR] bit
2. Delay at least 1 character period
3. Read URBRn again

ULSR[BI] will remain asserted for the duration of the break. The UART block will 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

JTAG5: The JTAG fails to capture the correct values of the receive pins of SerDes Interface

Description: The JTAG fails to capture the correct values of the receive pins of SerDes Interface as per IEEE 1149.1 standard.

Impact: Connectivity of these pins cannot be tested on boards.

Workaround: The affected RX pins will not be declared as inputs in the BSDL file. These pins are listed as type “linkage”. The rest of the part is testable with standard JTAG board interconnection tests.

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.