MPC8315ECE
Rev 4, 05/2014

# MPC8315E Chip Errata

This document details all known silicon errata for the MPC8315E, MPC8315, MPC8314E, and MPC8314. The following table provides a revision history for this document.

**Table 1. Document Revision History**

| Revision | Date | Significant Changes |
|---|---|---|
| 4 | 05/2014 | • Renamed SATA-A003 to A-005255<br>• Renamed USB38 to A-003817<br>• Added USB errata A-003837, A-003827, A-003829, and A-003845<br>• Added SATA errata A-005035, and A-005636<br>• Added eTSEC erratum A-007207<br>• Renamed eTSEC-A002 to A-006502 |
| 3 | 09/2011 | • Added USB-A007, CPU-A022, SATA-A003<br>• Updated USB-A001, PEX5 |
| 2 | 1/2011 | • Added eTSEC-A004, eLBC-A002, SEC-A001, USB-A003, USB-A005<br>• Updated SATA2, PEX7, General17, USB32, USB-A002 |
| 1 | 6/2010 | • Added eTSEC76, eTSEC78, eTSEC79, eTSEC-A001, eTSEC-A002, IEEE1588-A001, CPU-A002, SATA-A002, PCIe-A002, eLBC-A001, USB32, USB33, USB34, USB35, USB36, USB37, USB38, USB-A001, USB-A002 and USB20<br>• Updated eTSEC13, eTSEC38, eTSEC39, CPU6 and PEX7<br>• Removed IEEE 1588_13 because MPC8315 does not support TBI mode |
| 0 | 2/2009 | Initial public release |

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

## Table 2. Revision Level to Part Marking Cross-Reference

| Part Number | Device Marking | Processor Version Register (PVR) | System Version Register (SVR) | | |
|---|---|---|---|---|---|
| | Rev 1.0, Rev 1.1, and Rev 1.2 | Rev 1.0, Rev 1.1, and Rev 1.2 | Rev 1.0 | Rev 1.1 | Rev 1.2 |
| MPC8315E | M27K | 8085_0020 | 80B4_0010 | 80B4_0011 | 80B4_0012 |
| MPC8315 | M27K | 8085_0020 | 80B5_0010 | 80B5_0011 | 80B5_0012 |
| MPC8314E | M27K | 8085_0020 | 80B6_0010 | 80B6_0011 | 80B6_0012 |
| MPC8314 | M27K | 8085_0020 | 80B7_0010 | 80B7_0011 | 80B7_0012 |

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 3. Summary of Silicon Errata and Applicable Revision

| Errata | Name | Projected Solution | Silicon Rev. | | |
|---|---|---|---|---|---|
| | | | 1.0 | 1.1 | 1.2 |
| | **eTSEC** | | | | |
| eTSEC5 | WWR bit Anomaly | No plans to fix | Yes | Yes | Yes |
| eTSEC7 | RMCA, RBCA counters do not correctly count valid VLAN tagged frames | No plans to fix | Yes | Yes | Yes |
| eTSEC9 | Error in arbitrary extraction offset | No plans to fix | Yes | Yes | Yes |
| eTSEC13 | Fetches with errors not flagged, may cause livelock or false halt | No plans to fix | Yes | Yes | Yes |
| eTSEC18 | Parsing of MPLS label stack or non-IPv4/IPv6 label not supported | No plans to fix | Yes | Yes | Yes |
| eTSEC19 | Compound filer rules do not roll back the mask | No plans to fix | Yes | Yes | Yes |
| eTSEC20 | Filer does not support matching against broadcast address flag PID1[EBC] | No plans to fix | Yes | Yes | Yes |
| eTSEC21 | L3 fragment frame files on non-existent source/destination ports | No plans to fix | Yes | Yes | Yes |
| eTSEC22 | RxBD[TR] not asserted during truncation when last 4 bytes match CRC | No plans to fix | Yes | Yes | Yes |
| eTSEC24 | Parser results may be lost if TCP/UDP checksum checking is enabled | No plans to fix | Yes | Yes | Yes |
| eTSEC25 | Transmission of truncated frames may cause hang or lost data | No plans to fix | Yes | Yes | Yes |
| eTSEC28 | eTSEC does not support parsing of LLC/SNAP/VLAN packets | No plans to fix | Yes | Yes | Yes |
| eTSEC29 | Arbitrary extraction on short frames uses data from previous frame | No plans to fix | Yes | Yes | Yes |
| eTSEC30 | Preamble-only with error causes false CR error on next frame | No plans to fix | Yes | Yes | Yes |

*Table continues on the next page...*

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

| Errata | Name | Projected Solution | Silicon Rev. | | |
|---|---|---|---|---|---|
| | | | 1.0 | 1.1 | 1.2 |
| eTSEC31 | eTSEC filer reports incorrect Ether-types with certain MPLS frames | No plans to fix | Yes | Yes | Yes |
| eTSEC33 | Parser does not check VER/TYPE of PPPoE packets | No plans to fix | Yes | Yes | Yes |
| eTSEC34 | Some combinations of Tx packets may trigger a false Data Parity Error (DPE) | Fixed in Rev 1.1 | Yes | No | No |
| eTSEC36 | Generation of Ethernet pause frames may cause Tx lockup and false BD close | Fixed in Rev. 1.1 | Yes | No | No |
| eTSEC37 | eTSEC half duplex receiver packet corruption | No plans to fix | Yes | Yes | Yes |
| eTSEC38 | eTSEC Data Parity Error (DPE) does not abort transmit frames | Partial fix in Rev 1.1 | Yes | No | No |
| eTSEC39 | May drop Rx packets in non-FIFO modes with lossless flow control enabled | Fixed in Rev. 1.2 | Yes | Yes | No |
| eTSEC40 | Rx synchronization error may cause corrupted packets and limitations with Gigabit operation | Fixed in Rev1.1 | Yes | No | No |
| eTSEC41 | Multiple BD Tx frame may cause hang | No plans to fix | Yes | Yes | Yes |
| eTSEC42 | Frame is dropped with collision and HALFDUP[Excess Defer] = 0 | No plans to fix | Yes | Yes | Yes |
| eTSEC44 | No parser error for packets containing invalid IPv6 routing header packet | No plans to fix | Yes | Yes | Yes |
| eTSEC45 | eTSEC parser does not perform length integrity checks | No plans to fix | Yes | Yes | Yes |
| eTSEC46 | eTSEC does not verify IPv6 routing header type field | No plans to fix | Yes | Yes | Yes |
| eTSEC47 | No parse after back-to-back IPv6 routing header | No plans to fix | Yes | Yes | Yes |
| eTSEC48 | L4 info passed to filer in L2/L3-only mode | No plans to fix | Yes | Yes | Yes |
| eTSEC 52 | AC spec is not met for the higher nibble (D1) of data txd[0:3] | Fixed in Rev 1.1 | Yes | No | No |
| eTSEC53 | L2 arb extract shifted with shim headers not multiple of 4 bytes | No plans to fix | Yes | Yes | Yes |
| eTSEC54 | Frames greater than 9600 bytes with TOE = 1 will hang controller | No plans to fix | Yes | Yes | Yes |
| eTSEC55 | Arbitrary Extraction cannot extract last data bytes of frame | No plans to fix | Yes | Yes | Yes |
| eTSEC56 | Setting RCTRL[LFC] = 0 may not immediately disable LFC | No plans to fix | Yes | Yes | Yes |
| eTSEC58 | VLAN Insertion corrupts frame if user-defined Tx preamble enabled | No plans to fix | Yes | Yes | Yes |
| eTSEC59 | False parity error at Tx startup | No plans to fix | Yes | Yes | Yes |
| eTSEC60 | Tx data may be dropped at low system to Tx clock ratios | Fixed in Rev1.1 | Yes | No | No |
| eTSEC61 | Rx may hang if RxFIFO overflows | Fixed in Rev. 1.1 | Yes | No | No |
| eTSEC62 | Rx packet padding limitations at low clock ratios | No plans to fix | Yes | Yes | Yes |
| eTSEC63 | False TCP/UDP checksum error for some values of pseudo header Source Address | No plans to fix | Yes | Yes | Yes |
| eTSEC64 | User-defined Tx preamble incompatible with Tx Checksum | No plans to fix | Yes | Yes | Yes |
| eTSEC65 | Transmit fails to utilize 100% of line bandwidth | No plans to fix | Yes | Yes | Yes |
| eTSEC66 | Controller stops transmitting pause control frames | Fixed in Rev 1.2 | No | Yes | No |

*Table continues on the next page...*

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

| Errata | Name | Projected Solution | Silicon Rev. | | |
|---|---|---|---|---|---|
| | | | 1.0 | 1.1 | 1.2 |
| eTSEC67 | ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software | No plans to fix | Yes | Yes | Yes |
| eTSEC68 | Half-duplex collision on FCS of Short Frame may cause Tx lockup | Documentation update | Yes | Yes | Yes |
| eTSEC69 | Magic Packet Sequence Embedded in Partial Sequence Not Recognized | No plans to fix | Yes | Yes | Yes |
| eTSEC70 | MAC: Malformed Magic Packet Triggers Magic Packet Exit | No plans to fix | Yes | Yes | Yes |
| eTSEC71 | The value of TSEC_ID2 is incorrect | No plans to fix | Yes | Yes | Yes |
| eTSEC72 | Receive pause frame with PTV = 0 does not resume transmission | No plans to fix | Yes | Yes | Yes |
| eTSEC73 | TxBD polling loop latency is 1024 bit-times instead of 512 | No plans to fix | Yes | Yes | Yes |
| eTSEC76 | Excess delays when transmitting TOE=1 large frames | No plans to fix | Yes | Yes | Yes |
| eTSEC 78 | Controller may not be able to transmit pause frame during pause state | No plans to fix | Yes | Yes | Yes |
| eTSEC79 | Data corruption may occur in SGMII mode | No plans to fix | Yes | Yes | Yes |
| eTSEC-A001 | MAC: Pause time may be shorter than specified if transmit in progress | No plans to fix | Yes | Yes | Yes |
| A-006502 | Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame | No plans to fix | Yes | Yes | Yes |
| eTSEC-A004 | User-defined preamble not supported at low clock ratios | No plans to fix | Yes | Yes | Yes |
| A-007207 | TBI link status bit may stay up after SGMII electrical idle is detected | No plans to fix | Yes | Yes | Yes |
| **IEEE1588** | | | | | |
| IEEE 1588_2 | IEEE 1588 not supported in SGMII mode | No plans to fix | Yes | Yes | Yes |
| IEEE 1588_8 | Writing Offset registers during use may yield unpredictable results | No plans to fix | Yes | Yes | Yes |
| IEEE 1588_9 | 1588 reference clock limited to 1/2 controller core clock in asynchronous mode | No plans to fix | Yes | Yes | Yes |
| IEEE 1588_11 | 1588 reference clock pulse required between writes to TMR_ALARMn_L and TMR_ALARMn_H | No plans to fix | Yes | Yes | Yes |
| IEEE 1588_12 | 1588 alarm fires when programmed to less than current time | No plans to fix | Yes | Yes | Yes |
| IEEE 1588_15 | Use of asynchronous 1588 reference clock may cause errors | No plans to fix | Yes | Yes | Yes |
| IEEE 1588_16 | Odd prescale values not supported | No plans to fix | 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 |
| IEEE 1588_18 | FIPER's periodic pulse phase not realigned when the 1588 current time is adjusted | No plans to fix | Yes | Yes | Yes |
| IEEE 1588_21 | IEEE 1588 accuracy can be adversely impacted in systems using multiple unsynchronized gigabit Ethernet ports | No plans to fix | Yes | Yes | Yes |
| IEEE1588-A001 | Incorrect received timestamp or dropped packet when 1588 time-stamping is enabled | No plans to fix | Yes | Yes | Yes |

*Table continues on the next page...*

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

| Errata | Name | Projected Solution | Silicon Rev. | | |
|---|---|---|---|---|---|
| | | | 1.0 | 1.1 | 1.2 |
| | **PCI** | | | | |
| 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 |
| PCI19 | Dual-address cycle inbound write accesses can cause data corruption | No plans to fix | Yes | Yes | Yes |
| PCI20 | PCI controller may hang when returning from "PCI pins low" state | No plans to fix | Yes | Yes | Yes |
| PCI23 | Device locks up if a wake-up event occurs immediately after D3-warm instruction by e300 | Fixed in Rev 1.1. | Yes | No | No |
| PCI24 | While ICACHE on, critical data word access is corrupted | Fixed in Rev 1.1. | Yes | No | No |
| PCI 26 | PCI input setup to clock timing ($t_{PCIVKH}$) does not meet the specification | No plans to fix | Yes | Yes | Yes |
| | **DDR** | | | | |
| DDR21 | MCK/$\overline{\text{MCK}}$ AC differential crosspoint voltage outside JEDEC specifications | No plans to fix | Yes | Yes | Yes |
| DDR22 | DRAMs using 12 × 8 × 2 configurations cannot be used in 16-bit bus mode | No plans to fix | Yes | Yes | Yes |
| | **CPU** | | | | |
| CPU6 | DTLB LRU logic does not function correctly | No plans to fix | Yes | Yes | Yes |
| CPU-A002 | CPU may hang after load from cache-inhibited, unguarded memory | No plans to fix | Yes | Yes | Yes |
| CPU-A022 | The e300 core may hang while using critical interrupt | No plans to fix | Yes | Yes | Yes |
| | **DMA** | | | | |
| DMA2 | Data corruption by DMA when destination address hold (DAHE) bit is used | No plans to fix | Yes | Yes | Yes |
| | **SATA** | | | | |
| SATA2 | Reads of one sector from hard disk may return less data than requested | No plans to fix | Yes | Yes | Yes |
| SATA3 | SATA controller hangs when handling data integrity errors | No plans to fix | Yes | Yes | Yes |
| SATA4 | SATA BIST Activate FIS L bit is only valid in combination with the T bit | No plans to fix | Yes | Yes | Yes |
| SATA5 | SATA Host does not acknowledge BIST Activate FIS and it immediately enters loopback mode. | No plans to fix | Yes | Yes | Yes |
| SATA6 | BIST-L mode is not enabled by BIST-L FIS | No plans to fix | Yes | Yes | Yes |
| SATA7 | Transmit fails the spread spectrum clocking (SSC) modulation limits | No plans to fix | Yes | Yes | Yes |
| SATA8 | OOB Signal Detection Threshold failure | No plans to fix | Yes | Yes | Yes |
| SATA9 | Fails the SATA InterOperability Sinusoidal Jitter test | No plans to fix | Yes | Yes | Yes |
| SATA10 | Marginally fails Differential mode return loss test | No plans to fix | Yes | Yes | Yes |
| SATA11 | Spread Spectrum feature does not work as intended | No plans to fix | Yes | Yes | Yes |
| SATA12 | SATA: DMAT handling in SATA not as expected | No plans to fix | Yes | Yes | Yes |
| SATA-A002 | ATAPI commands may fail to complete | No plans to fix | Yes | Yes | Yes |

*Table continues on the next page...*

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

| Errata | Name | Projected Solution | Silicon Rev. | | |
|---|---|---|---|---|---|
| | | | 1.0 | 1.1 | 1.2 |
| A-005255 | Failure to detect single SYNC primitive | No plans to fix | Yes | Yes | Yes |
| A-005035 | Possible data loss if PRD[DBA] or PRD[DWC] is not at least 16-byte aligned | No plans to fix | Yes | Yes | Yes |
| A-005636 | Auto-activate feature enabled in DMA setup command causes timeout | No plans to fix | Yes | Yes | Yes |
| **PEX** | | | | | |
| PEX1 | No support of PCI Express completions with BCM bit set (PCIX bridge interface) | No plans to fix | Yes | Yes | Yes |
| PEX2 | DMA Interrupt descriptor race condition (IDRC) | No plans to fix | Yes | Yes | Yes |
| PEX5 | PCI Express LTSSM may fail to properly train with a link partner following HRESET# | No plans to fix | Yes | Yes | Yes |
| PEX6 | Hot Reset from RC to remote link causes RC configuration space registers to reset | Fixed in Rev 1.1 | Yes | No | No |
| PEX7 | Recovery from hot reset or link down | No plans to fix | Yes | Yes | Yes |
| PCIe-A002 | PCI Express Packets may be transmitted with excess data on x1 link | No plans to fix | Yes | Yes | Yes |
| **eLBC** | | | | | |
| eLBC2 | UPM does not have indication of completion of a Run Pattern special operation | No plans to fix | Yes | Yes | Yes |
| eLBC3 | eLBC NAND Flash memory has an ECC syndrome that collides with the JFFS2 marker in Linux | Fixed in Rev 1.1 | Yes | No | No |
| eLBC5 | LTEATR and LTEAR may show incorrect values under certain scenarios | No plans to fix | Yes | Yes | Yes |
| eLBC-A001 | Simultaneous FCM and GPCM or UPM operation may erroneously trigger bus monitor timeout | No plans to fix | Yes | Yes | Yes |
| eLBC-A002 | Core may hang while booting from NAND using FCM | No plans to fix | Yes | Yes | Yes |
| **General** | | | | | |
| General11 | DUART, JTAG, and TDM input high voltage does not meet the specification | No plans to fix | Yes | Yes | Yes |
| General12 | CSB deadlock | Fixed in Rev 1.1. | Yes | No | No |
| General 14 | Electrostatic Discharge (ESD) may fail to meet the 2KV Human body body model (HBM) | Fixed in Rev 1.2 | Yes | Yes | No |
| General15 | Electrostatic discharge (ESD) may fail to meet the 500 V charged device model (CDM) | No plans to fix | Yes | Yes | Yes |
| General16 | Enabling I$^2$C could cause I$^2$C bus freeze when other I$^2$C devices communicate | No plans to fix | Yes | Yes | Yes |
| General17 | DUART: Break detection triggered multiple times for a single break assertion | No plans to fix | Yes | Yes | Yes |
| JTAG5 | The JTAG fails to capture the correct values of the receive pins of SerDes Interface | No plans to fix | Yes | Yes | Yes |
| **RESET** | | | | | |
| Reset4 | Hard Coded Reset Configuration Word Option is not functional | Fixed in Rev 1.1 | Yes | No | No |

*Table continues on the next page...*

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

| Errata | Name | Projected Solution | Silicon Rev. | | |
|---|---|---|---|---|---|
| | | | 1.0 | 1.1 | 1.2 |
| | **SEC** | | | | |
| SEC9 | Protocol descriptor hang conditions | No plans to fix | Yes | Yes | Yes |
| SEC10 | AES-GCM IV Length Restriction | No plans to fix | Yes | Yes | Yes |
| SEC11 | Performance counter register access requirement | No plans to fix | Yes | Yes | Yes |
| SEC12 | TLS_SSL_Block_Inbound HMAC Error | No plans to fix | Yes | Yes | Yes |
| SEC14 | Non-compliant implementation of deterministic pseudo-random number generator | No plans to fix | Yes | Yes | Yes |
| SEC-A001 | Channel Hang with Zero Length Data | No plans to fix | Yes | Yes | Yes |
| | **USB** | | | | |
| USB15 | Read of PERIODICLISTBASE after successive writes may return a wrong value in host mode | No plans to fix | Yes | Yes | Yes |
| 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 |
| USB20 | Problem with configuration of RCWH/ CFG_RESET_SOURCE when ULPI intended | No plans to fix | Yes | Yes | Yes |
| USB21 | SE0_NAK issue | No plans to fix | Yes | Yes | Yes |
| USB23 | UTMI device mode (on-chip phy) issue in High speed | Fixed in Rev1.1 | Yes | 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 |
| USB26 | NackCnt field is not decremented when received NYET during FS/LS Bulk/Interrupt mode | No plans to fix | 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 |
| 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 | No plans to fix | Yes | Yes | Yes |
| USB29 | Priming ISO over SOF will cause transmitting bad packet with correct CRC | No plans to fix | Yes | Yes | Yes |
| USB30 | High speed output impedance fails marginally. | No plans to fix | Yes | Yes | Yes |
| USB31 | Transmit data loss based on bus latency | No plans to fix | Yes | Yes | Yes |
| USB32 | Missing SOFs and false babble error due to Rx FIFO overflow | No plans to fix | 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 |
| USB34 | NAK counter decremented after receiving a NYET from device | No plans to fix | Yes | Yes | Yes |
| USB35 | Core device fails when it receives two OUT transactions in a short time | No plans to fix | Yes | Yes | Yes |
| USB36 | CRC not inverted when host under-runs on OUT transactions | No plans to fix | Yes | Yes | Yes |

*Table continues on the next page...*

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

| Errata | Name | Projected Solution | Silicon Rev. | | |
|---|---|---|---|---|---|
| | | | 1.0 | 1.1 | 1.2 |
| USB37 | OTG Controller as Host does not support Data-line Pulsing Session Request Protocol | No plans to fix | Yes | Yes | Yes |
| A-003817 | USB Controller locks after Test mode "Test_K" is completed | No plans to fix | Yes | Yes | Yes |
| USB-A001 | Last read of the current dTD done after USB interrupt | No plans to fix | Yes | Yes | Yes |
| USB-A002 | Device does not respond to INs after receiving corrupted handshake from previous IN transaction | No plans to fix | Yes | Yes | Yes |
| USB-A003 | Illegal NOPID TX CMD issued by USB controller with ULPI interface | No plans to fix | Yes | Yes | Yes |
| USB-A005 | ULPI Viewport not Working for Read or Write Commands With Extended Address | No plans to fix | Yes | Yes | Yes |
| USB-A007 | Host controller fails to enter the PING state on timeout during High Speed Bulk OUT/DATA transaction | No plans to fix | Yes | Yes | Yes |
| A-003827 | DATA PID error interrupt issued twice for the same high bandwidth ISO transfer | No plans to fix | Yes | Yes | Yes |
| A-003837 | When operating in test mode, the CSC bit does not get set to 1 to indicate a change on CCS | No plans to fix | Yes | Yes | Yes |
| A-003829 | Host detects frame babble but does not halt the port or generate an interrupt | No plans to fix | Yes | Yes | Yes |
| A-003845 | Frame scheduling robustness-Host may issue token too close to uframe boundary | No plans to fix | Yes | Yes | Yes |

## eTSEC5: WWR bit Anomaly

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## eTSEC9: Error in arbitrary extraction offset

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## eTSEC13: Fetches with errors not flagged, may cause livelock or false halt

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## eTSEC19: Compound filer rules do not roll back the mask

**Description:** Devices: MPC8315E, MPC8314E

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 4. 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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No Plan to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

| | |
|---|---|
| **Description:** | Devices: MPC8315E, MPC8314E<br>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:** | No plans to fix |

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

**Description:** Devices: MPC8315E, MPC8314E

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 1.1

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

**Description:** Devices: MPC8315E, MPC8314E

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. 1.1

## eTSEC37: eTSEC half duplex receiver packet corruption

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** Partial fix in Rev 1.1

On detecting a Tx parity error, the controller does the following:

1. Abort the transmitting packet with an error, and all of the applicable:
   a. truncate frame without CRC appended in FIFO modes
   b. generate bad FCS in MAC modes
   c. assert TX_ER in MAC and GMII-style FIFO modes
   d. in some, but not all, cases, error code group in 16-bit encoded FIFO mode
2. Flush all active frames in the Tx FIFO and close all open BDs with an underrun error (TxBD[UN]=1). The controller may have up to 3 frames active in the Tx FIFO in addition to the frame with the parity error. The transmit buffer descriptor pointers (TBPTRn) may end up pointing to any of the up to 4 BDs open at the time the error occurred.
3. Halt all Tx queues
4. Set the TXE, DPE, and EBERR bits of the EVENT register. May also set IEVENT[XFUN].

Note: if the parity error occurs near the beginning of the frame, before frame transmission starts, the entire frame is discarded without being transmitted, and the TxBD closed with an underrun error.

In order to recover normal operation, software must do the following:

1. Clear the DPE, EBERR, XFUN and TXE bits in IEVENT
2. For each enabled ring, if the BD that the TBPTR*n* points to has the underrun bit set, rewind the TBPTRn back to the first BD with the underrun bit set.
3. Set each “rewound” BD with underrun back to ready state, including clearing the underrun and any other bits written by the controller.
4. Do an abbreviated soft reset of the controller.

   In MAC modes:

a. Clear MACCFG1[Tx_Flow]
b. Wait 256 TX_CLK cycles
c. Clear MACCFG1[Tx_EN]
d. Wait 3 TX_CLK cycles
e. Set MACCFG1[TX_EN] and, if desired, MACCFG1[Tx_Flow]

In FIFO modes:

a. Clear FIFOCFG[TXE]
b. Set FIFOCFG[TXE]

5. Clear all TSTAT[THLTn] bits

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

**Description:** Devices: MPC8315E, MPC8314E

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. 1.2

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

**Description:** Devices: MPC8315E, MPC8314E

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 Rev1.1

## eTSEC41: Multiple BD Tx frame may cause hang

**Description:** Devices: MPC8315E, MPC8314E

In section 15.6.7.2 of the MPC8315ERM (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:** No plan to fix

## eTSEC42: Frame is dropped with collision and HALFDUP[Excess Defer] = 0

**Description:** Devices: MPC8315E, MPC8314E

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

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

**Description:** Devices: MPC8315E, MPC8314E

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

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

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

The logic 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.

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

**Workaround:** Consistency checks for IPv6 routing header packets must be performed in software, or skipped. If a packet does have a valid IPv6 routing header, then L4 checksum result, if enabled, can be considered as valid. Otherwise, software should consider the packet as malformed and should not use the packet's L4 checksum result stored in RxFCB.

**Fix plan:** No plans to fix

## eTSEC45: eTSEC parser does not perform length integrity checks

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## eTSEC46: eTSEC does not verify IPv6 routing header type field

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plan to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## eTSEC 52: AC spec is not met for the higher nibble (D1) of data txd[0:3]

**Description:** Devices: MPC8315E, MPC8314E

In RGMII 1000 mode, where TXD switches at both the edges of the clock (TXC), AC spec is not met for the higher nibble (D1) of data txd[0:3]. Data corruption occurs because of a glitch, which causes data to become Zero. So the data valid time for the higher nibble (D1) is reduced by the width of the glitch.

For ETSEC1, glitch width = 1.4ns

ETSEC2. glitch width = 0.9ns

Effective data (D1) valid width for etsec1 = 4ns – 1.4ns = 2.6ns

Effective data (D1) valid width for etsec2 = 4ns – 0.9ns = 3.1ns

This may lead to AC specification violation at the PHY and incorrect data capture. Also, RGMII 1000 may not work correctly.

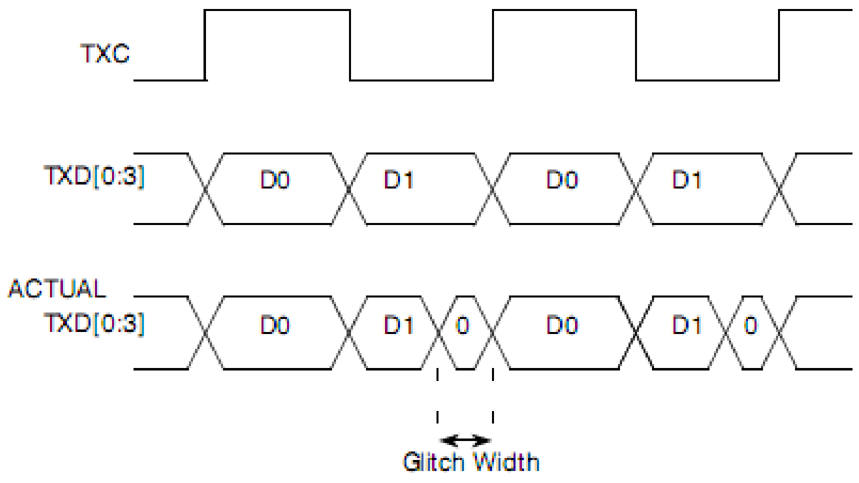

**Figure 1. RGMII AC Timing diagram**

**Impact:** RGMII mode may not be functional, if external RGMII PHY requires a tighter specification.

**Workaround:**

- Use an external RGMII PHY with a specification that meets the above restriction.

OR

- Use the SGMII interface for gigabit operation.

**Fix plan:** Fixed in Rev 1.1

## eTSEC53: L2 arb extract shifted with shim headers not multiple of 4 bytes

**Description:** Devices: MPC8315E, MPC8314E

L2 arbitrary extraction offsets >= 8 ordinarily start at the DA of the L2 header. If shim headers are enabled (RCTRL[L2OFF]! = 0), L2 arb extract offsets >= 8 start at the beginning of the shim header, and continue into the regular L2 header if the shim header is less than 56 bytes long. If the shim header length is not a multiple of 4 bytes, the controller does not take the byte shift into account when extracting from the regular L2 header and extracts the wrong bytes.

**Impact:** L2 arbitrary extraction is not supported if shim headers are enabled and they are not a multiple of 4 bytes in length.

**Workaround:**

- Option 1: Do not use shim headers (RCTRL[L2OFF] = 0).
- Option 2: Set the shim header length to multiples of 4 bytes.

**Fix plan:** No plan to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## eTSEC55: Arbitrary Extraction cannot extract last data bytes of frame

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## eTSEC58: VLAN Insertion corrupts frame if user-defined Tx preamble enabled

**Description:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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.

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:** Devices: MPC8315E, MPC8314E

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 Rev1.1

## eTSEC61: Rx may hang if RxFIFO overflows

**Description:** Devices: MPC8315E, MPC8314E

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. 1.1

## eTSEC62: Rx packet padding limitations at low clock ratios

**Description:** Devices: MPC8315E, MPC8314E

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:** No plan to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## eTSEC64: User-defined Tx preamble incompatible with Tx Checksum

**Description:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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

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

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

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

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

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

**Fix plan:** No Plan to fix

## eTSEC66: Controller stops transmitting pause control frames

**Description:** Devices: MPC8315E, MPC8314E

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 $(9608 + 20 + 72) \times 8 = 77,600$ bit times. For MAXFRM=1536 frames, that is $(1544+20+72) \times 8 = 13,088$ 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 1.2

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

**Description:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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

## eTSEC76: Excess delays when transmitting TOE=1 large frames

**Description:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8314E, MPC8315E

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: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Affects:** Ethernet

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

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

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

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

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

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

For affected systems, there is no 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_2: IEEE 1588 not supported in SGMII mode

**Description:** Devices: MPC8315E, MPC8314E

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:** No plan to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plan to fix.

## IEEE 1588_12: 1588 alarm fires when programmed to less than current time

**Description:** Devices: MPC8315E, MPC8314E

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 TSEC_TMR_ALARMn_H to arm the event.

**Fix plan:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plan to fix.

## IEEE 1588_16: Odd prescale values not supported

**Description:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## IEEE 1588_21: IEEE 1588 accuracy can be adversely impacted in systems using multiple unsynchronized gigabit Ethernet ports

**Description:** Devices: MPC8315E, MPC8314E

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

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

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

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

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

**Fix plan:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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 or data corruption may occur without error indication.

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

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

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

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

**Fix plan:** No plans to fix

## 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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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

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

**Description:** Devices: MPC8315E, MPC8314E

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 1.1.

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

**Description:** Devices: MPC8315E, MPC8314E

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 1.1.

## PCI 26: PCI input setup to clock timing ($t_{PCIVKH}$) does not meet the specification

**Description:** Devices: MPC8315E, MPC8314E

PCI input setup to clock timing may not meet the specification at 66MHz. The $t_{PCIVKH}$ minimum value is 3.3ns instead of 3.0ns.

**Impact:** The board design should take into account the input setup timing violation.

**Workaround:** None.

**Fix plan:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:

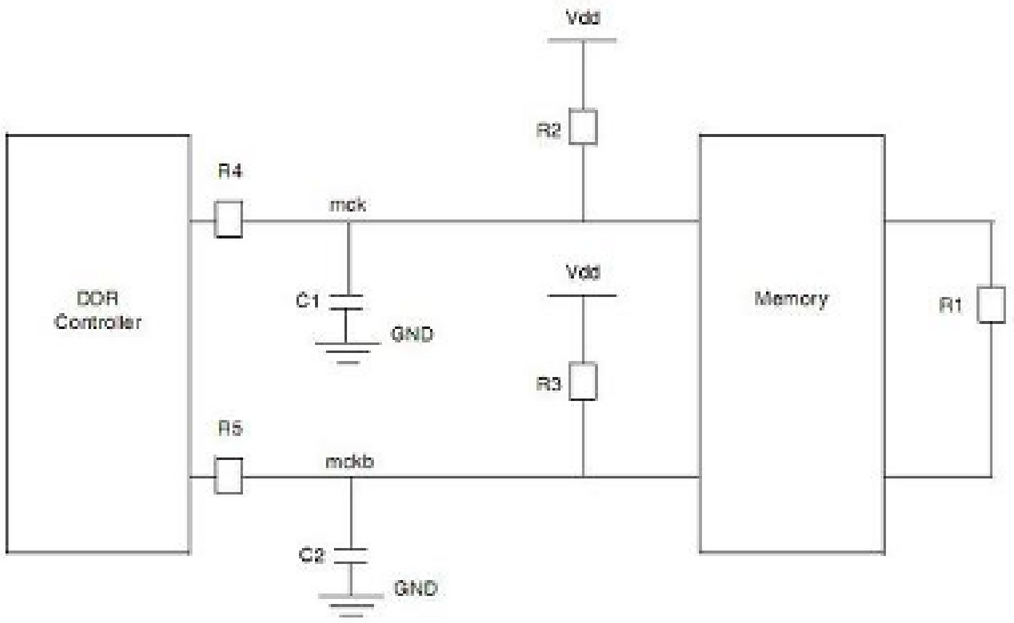

**Figure 2. Recommended connections for DDR controller and memory**

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.

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:** Devices: MPC8315E, MPC8314E

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

## CPU6: DTLB LRU logic does not function correctly

**Description:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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

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

**Description:** Devices: MPC8315E, MPC8314E

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

## SATA2: Reads of one sector from hard disk may return less data than requested

**Description:** Devices: MPC8315E

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

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

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

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

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

In this, there are two situations:

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

**Fix plan:** No plans to fix

## SATA3: SATA controller hangs when handling data integrity errors

**Description:** Devices: MPC8315E

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

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

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

**Workaround:** None

**Fix plan:** No plans to fix

## SATA4: SATA BIST Activate FIS L bit is only valid in combination with the T bit

**Description:** Devices: MPC8315E

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

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

**Workaround:** Use one of the following options:

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

**Fix plan:** No plans to fix

## SATA5: SATA Host does not acknowledge BIST Activate FIS and it immediately enters loopback mode.

**Description:** Devices: MPC8315E

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

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

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

**Fix plan:** No plans to fix

## SATA6: BIST-L mode is not enabled by BIST-L FIS

**Description:** Devices: MPC8315E

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

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

**Workaround:** Use the internal PHY control register to put SATA controller in BIST-L mode.

In order to configure the PHY Control Register, software must do the following:

1. Set the SATA in BIST-L mode by writing 0x0000_2902 to offset 0x15C (PHY Control Configuration Register1). Note that the value is shown from the register perspective (byte-swapped).
2. Enable the SATA controller by writing 0x8000_0000 to offset 0x02C (Host Control Register). Note that the value is shown from the register perspective (byte-swapped).

**Fix plan:** No plans to fix

## SATA7: Transmit fails the spread spectrum clocking (SSC) modulation limits

**Description:** Devices: MPC8315E

SATA standard provides an optional SSC modulation on transmit with limits of - 5000PPM. The 8315 SATA PHY will over modulate the clock resulting in the modulated frequency going below the standard limits of -5000ppm.

**Impact:** This overmodulation might result in receiver errors depending on the extra tolerance with the far end Rx path.

**Workaround:** Do not use the Spread Spectrum Modulation option on the Transmit side.

**Fix plan:** No plans to fix

## SATA8: OOB Signal Detection Threshold failure

**Description:** Devices: MPC8315E

The 8315 SATA PHY employs digital methods to detect OOB. It can correctly detect OOB sequences at a voltage level below that of the minimum as defined in the SATA standard.

**Impact:** The detection of OOB at lower voltages means that the device can detect and begin to negotiate the attached SATA devices in voltage levels that are below the SATA Standard.

**Workaround:** None.

**Fix plan:** No plans to fix

## SATA9: Fails the SATA InterOperability Sinusoidal Jitter test

**Description:** Devices: MPC8315E

SATA standard does not provide any specifics on Sinusoidal Jitter testing.

Sinusoidal Jitter test interpretation at several specific frequencies was added only to the SATA InterOp testing revisions 1.2 and above. The SATA ports will not be error free when presented with high levels of sinusoidal Jitter done in the SATA InterOp testing.

**Impact:** The SATA port on this device is not compliant to the SATA InterOperability testing revisions 1.2 and higher.

**Workaround:** None

**Fix plan:** No plans to fix

## SATA10: Marginally fails Differential mode return loss test

**Description:** Devices: MPC8315E

SATA standard specify the minimum return loss (SD11) for different frequency ranges. This measurement is highly dependent on the board layout and component positions. However, under good board layout and component position conditions the differential mode return loss test may still fail.

**Impact:** The marginal failure of the differential mode return loss test has not shown to be of any significance in practical applications. However, this device my not pass the SATA logo testing.

**Workaround:** None.

**Fix plan:** No plans to fix

## SATA11: Spread Spectrum feature does not work as intended

**Description:** Devices: MPC8315E

SATA PHY should be able to recover SSC modulated Data (downspread from -5700ppm to +700 ppm). However 8315 SATA PHY has a much less tracking range when Rx_SSC is enabled at Gen2 Speed.

**Impact:** Disparity and Code Violation errors might be observed on return path when Spread Spectrum is enabled or with SSC enabled Hard Disk.

**Workaround:** Disable Spread Spectrum option on the Receive side at 3Gbps.

**Fix plan:** No plans to fix

## SATA12: SATA: DMAT handling in SATA not as expected

**Description:** Devices: MPC8315E

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

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

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

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

**Fix plan:** No plans to fix

## SATA-A002: ATAPI commands may fail to complete

**Description:** Devices: MPC8315E

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

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

This is because SATA controller misassigns the command number

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

**Workaround:** For drive connected directly to SATA port:

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

For drives connected through PM Port:

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

**Fix plan:** No plans to fix

## A-005255: Failure to detect single SYNC primitive

**Affects:** SATA

**Description:** Devices: MPC8315E

When a single SYNC primitive is sent between the WTRM and XRDY primitives, the host controller may fail to detect this primitive. This can cause the host controller link state machine to remain in the "ending" state. As a result, the controller will not be able to return to the "idle“ state, will not respond to the XRDY primitive stream and will not set corresponding command completion event.

**Impact:** Read command issued by the controller may not complete if the device sends a single SYNC primitive.

**Workaround:** There is no workaround. If the device sends a single SYNC primitive and the host controller fails to detect it, the software will time out waiting for command completion event. To recover, the software can re-initialize the SATA Link by clearing HCONTROL[HC_ON] to 0x0 and then setting it to 0x1.

**Fix plan:** No plans to fix

## A-005035: Possible data loss if PRD[DBA] or PRD[DWC] is not at least 16-byte aligned

**Affects:** SATA

**Description:** SATA controller by design always tries to reach 64 byte alignment. If the PRD[DBA] or PRD[DWC] is not 64 byte aligned, then in the beginning or the end of transaction, it has to use 32-byte, 16-byte, 4-byte size transaction.

**Impact:** SATA controller may lose data when it tries to do 4-byte transaction.

**Workaround:** Software must make sure both PRD[DBA] and PRD[DWC] are at least 16-byte aligned  so that the SATA controller would avoid 4-byte access.

**Fix plan:** No plans to fix

## A-005636: Auto-activate feature enabled in DMA setup command causes timeout

**Affects:** SATA

**Description:** When NCQ is enabled, the SATA controller does not support DMA setup FIS with auto-activate enabled from the device. The SATA host may timeout without finishing the transaction.

**Impact:** This will have a minor performance impact as disabling the auto-activate feature requires the device to send a DMA setup as well as a DMA activate FIS to enable reception of the first data FIS.

**Workaround:** Software can work around this with one of the following options:

- Disable the DMA setup auto-activate feature by a set features command.
- Or, disable NCQ by setting the queue depth to one.

**Fix plan:** No plans to fix

## PEX1: No support of PCI Express completions with BCM bit set (PCIX bridge interface)

**Description:** Devices: MPC8315E, MPC8314E

To satisfy certain PCI-X protocol constraints, a PCI-X Bridge or PCI-X Completer for a PCI-X burst read in some cases will set the Byte Count field in the first PCI-X transaction of the Split Completion sequence to indicate the size of just that first transaction instead of the entire burst read. When this occurs, the PCI-X Bridge/PCI-X Completer will also set the BCM bit in that first PCI-X transaction, to indicate that the Byte Count field has been modified from its normal usage. A PCI Express Memory Read Requester needs to correctly handle the case when a PCI-X Bridge/PCI-X Completer sets the BCM bit. PCI Express Completers will never set the BCM bit.

The device does not expect the BCM to be set. A packet received with the BCM bit will cause unexpected behavior.

**Impact:** The device does not support the BCM behavior described in the PCI Express standard, and may not support PCI Express to PCI-X bridges.

**Workaround:** None

**Fix plan:** No plans to fix

## PEX2: DMA Interrupt descriptor race condition (IDRC)

**Description:** Devices: MPC8315E, MPC8314E

The DMA DONE bit is set and issues an interrupt before the buffer descriptor (BD) is written to memory.

**Impact:** After detecting the interrupt, the software may read wrong data from the descriptor status.

**Workaround:** When serving the interrupt, Software must check that the done bit in the chain descriptor is set. If it is not set, software should poll the DONE bit until it is set.

**Fix plan:** No plans to fix

## PEX5: PCI Express LTSSM may fail to properly train with a link partner following HRESET#

**Description:** Devices: MPC8315E, MPC8314E

Following HRESET#, the PCI Express controller will enter the internal LTSSM (Link Training and Status State Machine), and may fail to properly detect a receiver as defined in the PCI Express Base Specification. Failing to properly detect a receiver can be determined by reading the LTSSM State Status Registers (offset 0x404) in the PCI Express extended configuration space. When this failure has occurred the status code can be either 0 or 1h, indicating that it is still in a detect state. If the link has properly trained, the status code will read 0x16h.

**Impact:** Following HRESET#, (or a hot swap and/or dynamic power up/down on the link partner) the PCI Express controller may fail to properly train with an active link partner, causing the PCI Express controller to hang.

**Workaround:** During PEX initialization sequence, once the serdes reset sequence is over (RDONE bit is set in Serdes Reset Control Register - IMMRBAR offset 0xE_3020), wait for 1 ms before bringing the pex controller out of reset (setting the link reset, pab reset and csr reset bits in pex control register 1/2 - IMMRBAR offset 0x0_0140/0x0_0144)

**Fix plan:** No Plans to Fix

## PEX6: Hot Reset from RC to remote link causes RC configuration space registers to reset

**Description:** Devices: MPC8315E, MPC8314E

When the PCI Express is configured as a Root Complex, it may issue a hot reset to a remote link partner. This operation causes the Root Complex to reset its own configuration space registers.

**Impact:** The content of the RC configuration space registers, including any information gathered during the system configuration, is lost. This behavior may be incompliant with PCI Express software that assumes the content of the RC configuration space registers is maintained when issuing a hot reset.

**Workaround:** The software needs to maintain a copy of the relevant information outside the PCI Express configuration space registers area, and reinitialize the configuration registers after a hot reset is issued.

**Fix plan:** Fixed in Rev 1.1

## PEX7: Recovery from hot reset or link down

**Description:** Devices: MPC8315E, MPC8314E

The PCI Express CSB bridge enters a suspend mode as a result of a hot reset or link down event in either endpoint (EP) or root complex (RC) mode. When this happens, the PCI Express controller flushes all outstanding CSB transactions and does not accept new inbound transactions involving CSB and ATMU translation until the CSB bridge is unlocked, even though the link is automatically retrained properly and is stable in L0.

The CSB bridge cannot be unlocked without a soft reset. Software running on the local CPU must wait until the PCI Express CSB bridge is idle before performing a soft reset as well as re-enabling and reconfiguring the CSB bridge registers.

Further, even in RC mode, the PCI configuration space also gets reset as a result of a link down event (and not a hot reset event). This space needs to be reprogrammed during recovery.

A link down scenario can be caused by various reasons, and could occur even during the link negotiation phase of the initial link training coming out of a power-on reset. In this case, the LTSSM can temporarily go to L0, re-train after a quick link down, and then reach a stable link-up L0 state.

**Note**: If the MPC8315E/MPC8314E initiated the link recovery, this is not considered a link down event and the PCI Express CSB bridge does not enter suspend mode. In this case, the device is fully operational when the link is back to L0 state and there is no need for a soft reset.

**Impact:** When the system is in EP mode, the system will fail if the link goes up and the RC issues transactions involving CSB and ATMU translation to the endpoint before the PCI Express CSB bridge is unlocked.

Additionally, reconfiguring the CSB bridge registers and PCI configuration space (when in RC mode during a link down event) can complicate and slow down the software.

Finally, standard software drivers (such as Linux) do not expect that the PCI Express interface to be locked upon link retraining and therefore require special consideration.

**Workaround:** To recover from the above mentioned conditions, the following steps should be taken by the software running on the device.

When in EP mode:

1. In order to be interrupted when detecting the hot reset or link down, set the PEX_CSMIER[RSTIE] bit.
2. When the controller identifies a hot reset or link down event, the PEX_CSMISR[RST] bit gets set, and an interrupt is generated if the PEX_CSMIER[RSTIE] bit was set.
3. The software should immediately clear the CFG_READY bit.
4. Write one to clear PEX_CSMISR[RST].
5. Poll IBPIORP and WDMARP bits in PEX_CSB_STAT register (0x81c) to complete all outstanding transactions.
6. Perform a soft reset to the PCI Express CSB bridge by clearing and re-setting the PECRn[CBRST] bit (100ns between clearing and re-setting is sufficient).
7. Reset to the PCI Express CSR space by clearing and re-setting the PECRn[CSR_RST] bit (100ns between clearing and re-setting is sufficient).
8. Reprogram the PCI Express CSB bridge registers. All configuration register bits that are non-sticky are reset and need to be reprogrammed (offset 0x800-0xFFC).
9. Set the CFG_READY bit.

When in RC mode after a link down event occurs:

1. In order to be interrupted when detecting the link down, set the PEX_CSMIER[RSTIE] bit.
2. When the controller identifies a link down event, the PEX_CSMISR[RST] bit gets set, and an interrupt is generated if the PEX_CSMIER[RSTIE] bit was set. Note that this bit is not set for a hot reset event when the MPC8315E/MPC8314E is in RC mode.
3. Write one to clear PEX_CSMISR[RST].
4. Poll IBPIORP and WDMARP bits in PEX_CSB_STAT register (0x81c) to complete all outstanding transactions.
5. Perform a soft reset to the PCI Express CSB bridge by clearing and re-setting the PECRn[CBRST] bit (100ns between clearing and re-setting is sufficient).
6. Reprogram the PCI Express CSB bridge registers. All configuration register bits that are non-sticky are reset and need to be reprogrammed (offset 0x800-0xFFC).
7. Reconfigure the PCI configuration space.

When in RC mode after a hot reset event occurs:

1. Poll IBPIORP and WDMARP bits in PEX_CSB_STAT register (0x81c) to complete all outstanding transactions.
2. Perform a soft reset to the PCI Express CSB bridge by clearing and re-setting the PECRn[CBRST] bit (100ns between clearing and re-setting is sufficient).
3. Reprogram the PCI Express CSB bridge registers. All configuration register bits that are non-sticky are reset and need to be reprogrammed (offset 0x800-0xFFC).

The PCI Express controller is now ready for normal transactions.

**Fix plan:** No plans to fix

## PCIe-A002: PCI Express Packets may be transmitted with excess data on x1 link

**Description:** Devices: MPC8315E, MPC8314E

An internal error in the PCI Express controller may cause 8 bytes of packet header or data payload to be duplicated on transmit. Depending on the location of the duplicate bytes, this may cause a malformed packet error or CRC error detected in the link partner.

**Impact:** A PCI Express link partner may see excess correctable errors or malformed packets in x1 links.

Note that any PCI Express specification-compliant recipient at the ultimate end of the transaction may only recognize the erroneous packet as a Bad TLP and flag the error as correctable due to the LCRC error. It should respond with a NAK and then wait for the device to re-try the packet. The ultimate recipient should not recognize the erroneous packet as a Malformed TLP, since before reaching the transaction layer the Bad TLP should have been discarded by the ultimate recipient's link layer.

**Workaround:** None

**Fix plan:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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 1.1

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

| | |
|---|---|
| **Description:** | Devices: MPC8315E, MPC8314E<br><br>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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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

## General11: DUART, JTAG, and TDM input high voltage does not meet the specification

**Description:** Devices: MPC8315E, MPC8314E

The DUART, JTAG, and TDM 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:** Devices: MPC8315E, MPC8314E

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 1.1

## General 14: Electrostatic Discharge (ESD) may fail to meet the 2KV Human body body model (HBM)

**Description:** Devices: MPC8315E, MPC8314E

HBM (Human Body Model) ESD testing has shown that some USB and SGMII / PCIe PHY pins do not meet the 2kV HBM ESD criteria. HBM passes at 1kV. The following pins are impacted:

USB_DP, USB_DM, USB_RBIAS, USB_VBUS, USB_VDDA, USB_VDDA_BIAS, SD_PLL_TPA_ANA, TXA, $\overline{\text{TXA}}$, TXB, $\overline{\text{TXB}}$, SD_IMP_CAL_TX, SDAVDD, XPADVDD, XCOREVDD

**Impact:** Excessive ESD can cause damage to the device.

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

**Fix plan:** Fixed in Rev 1.2

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

**Description:** Devices: MPC8315E, MPC8314E

CDM (charged device model) ESD testing has shown that some pins, in particular the high speed SGMII/PCIe 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 I$^2$C could cause I$^2$C bus freeze when other I$^2$C devices communicate

**Description:** Devices: MPC8315E, MPC8314E

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

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

**Workaround:** Use one of the following workarounds:

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

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

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

**Fix plan:** No plans to fix

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

**Description:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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

## Reset4: Hard Coded Reset Configuration Word Option is not functional

**Description:** Devices: MPC8315E, MPC8314E

If the user sets the Hard Coded Reset Configuration Word Option (that is, CFG_RESET_SOURCE[0:3] is configured to 10xx or 1100), coherent system bus frequency and DDR controller frequency can go out of range as per specification.

**Impact:** System operation is not guaranteed for the hard-coded reset configuration word option.

**Workaround:**

- Do not use the hard-coded reset configuration word soft reset at all. Load the reset configuration word from I$^2$C EEPROM or LBC (NOR/NAND Flash interface) to specify the required setting.
- Use the I$^2$C boot sequencer EEPROM to disable the CFG_LOCK bit of the PCI Func. Configuration Register (which is at address 0x44 of the PCI Configuration space).

**Fix plan:** Fixed in Rev 1.1

## SEC9: Protocol descriptor hang conditions

**Description:** Devices: MPC8315E, MPC8314E

The SEC is designed to perform single pass encryption and integrity checking as required for security protocols including IPSec and SSL/TLS. When using the descriptor types required for single pass IPSEC and SSL/TLS, where a primary EU is selected, and no secondary EU is selected, the channel hangs rather than producing an illegal descriptor header error.

**Impact:** Channel hangs if no secondary EU is selected for single pass IPSEC and SSL/TLS. There are two specifically observed hang conditions, as follows:

1. IPSec inbound without a secondary EU (Outbound, with or without a secondary EU is fine; inbound with a secondary EU is fine). The channel behavior is not predictable and may hang.
2. TLS_SSL_STREAM inbound if the primary EU is AFEU and there is No secondary EU, the channel hangs.

**Workaround:** Single pass protocol descriptors are designed for simultaneous encryption and integrity checking, and operation which required two EUs. For IPSec or SSL with integrity only, other descriptor types should be used. Note that this errata does not impact IPSec with AES-GCM (a use of IPSec with a single primary EU). IPSec with AES-GCM uses a different descriptor type.

**Fix plan:** No plans to fix

## SEC10: AES-GCM IV Length Restriction

**Description:** Devices: MPC8315E, MPC8314E

Like most other modes of AES, AES-GCM uses a key and an IV to transform plaintext to ciphertext. The length of the IV is standardized by multiple independent bodies. The base AES-GCM specification allows the use of any size of IV or AAD or plaintext/ciphertext.

Other standards bodies, including the IEEE and IETF, define security protocols which use AES-GCM, such as MACSec (IEEE Std 802.1ae™-2005), and IPSec. The specifications for the use of AES-GCM within these security protocols restricts the size of the IV, AAD, and plaintext/ciphertext.

The AESU in the SEC performs AES-GCM without any issue whenever the IV is 96 bits, however the AESU fails under the following conditions:

1. The IV length is not 96 bits.
2. The initial IV Ghash operation produces a value with the 32 least significant bits all 1’s

**Impact:** Both IPSec and MACSec (IEEE 802.1ae) specify an IV length of 96 bits, which allows the SEC to support these protocols without restriction. The only known concern associated with this errata comes from the current draft standard for P1619.1 (Draft Standard for Authenticated Encryption with Length Expansion for Storage Devices), which permits IVs other than 96 bits in length. Because there is no way to predict whether the initial IV Ghash operation produces a value with the 32 least significant bits all 1’s, the SEC should not be used for AES-GCM in P1619.1, or any other protocol which does not limit IV length at 96 bits.

**Workaround:** For protocols specifying 96b IVs, no work around is necessary. For protocols specifying other IV lengths, none is possible.

**Fix plan:** No plans to fix

## SEC11: Performance counter register access requirement

**Description:** Devices: MPC8315E, MPC8314E

The SEC Poly Channel contains four registers that allow software to quickly obtain statistics related to SEC performance. All four of these registers can only be read with a single 64-bit read.

**Table 5. SEC Poly Channel Registers**

| |
|---|
| Fetch FIFO Enqueue Count |
| Descriptor Finished Count |
| Data Bytes In Count |
| Data Bytes Out Count |

**Impact:** Separate 32-bit reads of the upper and lower portions of the registers will cause the operation to fail.

**Workaround:** When accessing these registers, only use 64-bit reads.

**Fix plan:** No plans to fix

## SEC12: TLS_SSL_Block_Inbound HMAC Error

**Description:** Devices: MPC8315E, MPC8314E

The SSL/TLS protocol allows for the use of several different cipher suites to encrypt and integrity check payload data. The encryption cipher can be a stream cipher like RC-4, or a block cipher like 3DES or AES.

The SSL/TLS processing steps for out-bound operations with a block cipher are as follows:

1. Calculate HMAC over the SSL record header and payload.
2. Add HMAC to the end of the payload.
3. Calculate the number of bytes of payload and HMAC, plus a Pad Length byte, and add padding bytes to the record until the total number of bytes (payload||HMAC||padding||Pad Length) is an integral number of cipher blocks.
4. Encrypt the record, starting at the first byte of payload, ending at the Pad Length (Pad Length is included in the encryption).
5. Change the Record Header Length field to match the length of payload||HMAC||padding|| Pad Length

This order of operations is opposite to most other security protocols, but does not present any special challenges to single pass hardware accelerators.

For inbound, the order of operations is almost the opposite:

1. Decrypt the record, starting at the first byte of payload, ending at the Pad Length (Pad Length is included in the decryption).
2. Calculate the number of bytes of payload by subtracting the length of the HMAC, padding, and pad length from the Length field received with the record header.
3. Change the length field to match the payload length only. Remove the padding and pad length bytes.
4. Calculate HMAC over the SSL record header and decrypted payload.
5. Compare the received HMAC with the calculated HMAC, and if different, drop the record.

This order of operations presents a significant challenge to single pass hardware. It isn't possible to change the length field in the record header until the Pad Length byte has been decrypted, and if the length field isn't changed prior to inbound integrity checking, the calculated HMAC are wrong. The SEC implementation did not comprehend that the Length field would not be modified by software prior to launching the descriptor, as is done for out-bound processing.

**Impact:** TLS_SSL_Block_Inbound operations produce incorrect HMAC values.

**Workaround:**

- Option 1: For TLS_SSL_Outbound operations, use the SEC's single pass descriptor type 1000_1 for maximum performance. For TLS_SSL_Inbound, use two descriptors:
  - First descriptor: Use type 0001_0 'common non-snoop', with header set for the appropriate block cipher algorithm, to decrypt the payload through Pad Length. Upon completion of this descriptor, software changes the value of the Length field in the record header and launches the second descriptor.
  - Second Descriptor: Type 0001_0 'common non-snoop' set for the appropriate HMAC algorithm (for example, HMAC-SHA-1). The pointers in this descriptor tell the SEC to calculate an HMAC over the SSL Record Header||Payload. The SEC can be set to automatically compare the generated HMAC against the received HMAC, or the descriptor can write the generated HMAC to memory for a comparison by software.

- Option 2: For TLS_SSL_Outbound operations, use the SEC's single pass descriptor type 1000_1 for maximum performance. For TLS_SSL_Inbound, use a two-descriptor method that minimizes the total amount of data the SEC must read to generate a decrypted and authenticated record:
  - First descriptor: Use type 0001_0 'common non-snoop', with header set for the appropriate block cipher algorithm. Assuming CBC mode, decrypt the final block of the record, using as the IV the second-to-last block of the record. Do not write the decrypted data back to the memory location of the record. The purpose of decrypting the final block is to recover the Pad Length field, which is the last byte of the output generated by this descriptor. After you have the Pad Length, discard the decrypted data. Upon completion of this descriptor, software subtracts the length of the HMAC (a known constant value), padding, and pad length from the Length field, updates the Length field in the record header and launches the second descriptor.
  - Second Descriptor: Type 1000_1 TLS_SSL_Block. Because the Length field is set properly, the single pass TLS_SSL_Inbound descriptor properly decrypts and authenticates the record.

**Fix plan:** No plans to fix

## SEC14: Non-compliant implementation of deterministic pseudo-random number generator

**Description:** Devices: MPC8314E, MPC8315E

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

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

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

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

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

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

**Fix plan:** No plans to fix

## SEC-A001: Channel Hang with Zero Length Data

**Description:** Devices: MPC8315E, MPC8314E

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

## USB15: Read of PERIODICLISTBASE after successive writes may return a wrong value in host mode

**Description:** Devices: MPC8315E, MPC8314E

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

## 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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** No plans to fix

## USB23: UTMI device mode (on-chip phy) issue in High speed

**Description:** Devices: MPC8315E, MPC8314E

When used in UTMI device mode (on-chip phy), the proper chirp sequence is not generated in High speed mode.

**Impact:** USB in UTMI device mode (on-chip phy) does not work in High speed mode. However, there are no issues when USB is operating in UTMI Host mode.

**Workaround:** Use ULPI interface with external PHY for USB device mode in High speed mode.

**Fix plan:** Fixed in Rev1.1

## 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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8314E, MPC8315E

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: MPC8315E, MPC8314E

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: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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: MPC8315E, MPC8314E

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: MPC8315E, MPC8314E

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: MPC8315E, MPC8314E

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: MPC8315E, MPC8314E

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: MPC8315E, MPC8314E

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: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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:** Devices: MPC8315E, MPC8314E

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-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-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-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

***How to Reach Us:***

**Home Page:**
freescale.com

**Web Support:**
freescale.com/support