MPC8308 Chip Errata

This document details all known silicon errata for the MPC8308. The following table provides a revision history for this document.

Table 1. Document Revision History

Revision	Date	Significant Changes
5	09/2015	Added the following errata: • eSDHC A-005055 and A-009204 • IEEE1588 A-007734 • PCIe PEX5 • USB A-003845 Added IEEE 1588_19 to Table 3, Summary of Silicon Errata and Applicable Revision. Modified eTSEC69. Renamed eTSEC-A002 to A-006502 and modified the write-up. Renamed "PEX" heading to "PCIe". Re-ordered the USB errata numerically.
4	07/2013	Renamed USB38 to USB erratum A-003817 Added the following errata: • USB errata: A-003837, A-003827, A-003829 • eSDHC erratum: A-004577 Removed erratum DDR20
3	05/2012	• In Table 2, added silicon revision 1.1 SVR and PVR information. • In Table 3, added silicon revision 1.1 column. • In Table 3, updated the projected solution for the following errata: IEEE1588-A001, PCIe- A002, and A-003985.
2	12/2011	• Added eLBC-A002 • Removed eLBC1, eLBC2, eTSEC29, IEEE1588_8, USB21
1	11/2011	• Added USB-A005, USB-A007, CPU-A022, PCIe-A002, A-003985 • Removed SPI6 • Updated USB-A001
0	07/2010	Initial release

BUILT ON Power

freescale

© 2015 Freescale Semiconductor, Inc.

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

Table 2. Revision Level to Part Marking Cross-Reference

Part	Revision	Processor Version Register Value	System Version Register Value	Mask
MPC8308	1.1	8085_0020	8101_0111	M17Y
MPC8308	1.0	8085_0020	8101_0110	M17Y

Table 3 summarizes all known errata.

Table 3. Summary of Silicon Errata and Applicable Revision

Errata	Name	Projected Solution	Silicon Rev.	
			1.0	1.1
CPU				
CPU6	DTLB LRU logic does not function correctly	No plans to fix	Yes	Yes
CPU-A002	CPU may hang after load from cache-inhibited, unguarded memory	No plans to fix	Yes	Yes
CPU-A022	The e300 core may hang while using critical interrupt	No plans to fix	Yes	Yes
DDR				
DDR21	MCK/$\overline{\text{MCK}}$ AC differential crosspoint voltage outside JEDEC specifications	No plans to fix	Yes	Yes
DDR23	DDR read failure due to t_{DISKEW} spec violation	No plans to fix	Yes	Yes
eLBC				
eLBC5	LTEATR and LTEAR may show incorrect values under certain scenarios	No plans to fix	Yes	Yes
eLBC-A001	Simultaneous FCM and GPCM or UPM operation may erroneously trigger bus monitor timeout	No plans to fix	Yes	Yes
eLBC-A002	Core may hang while booting from NAND using FCM	No plans to fix	Yes	Yes
eSDHC				
eSDHC16	Manual Asynchronous CMD12 abort operation causes protocol violations	No plans to fix	Yes	Yes
eSDHC23	CMD CRC error or CMD index error may be set for CMD without data while CMD with data is in progress	No plans to fix	Yes	Yes
eSDHC-A002	BLKATTR[BLKCNT] does not return to 0 after Transfer Complete for multi-block transfer	No plans to fix	Yes	Yes
A-004577	PRSSTAT[DLA] bit does not reflect the data line state when any command with busy (R1b) is issued	No plans to fix	Yes	Yes
A-005055	A glitch is generated on the card clock with software reset or a clock divider change	No plans to fix	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
A-009204	System bus may hang if software register SYSCTL[RSTD] is set while DMA transactions are active	No plans to fix	Yes	Yes
eTSEC				
eTSEC42	Frame is dropped with collision and HALFDUP[Excess Defer] = 0	No plans to fix	Yes	Yes
eTSEC58	VLAN Insertion corrupts frame if user-defined Tx preamble enabled	No plans to fix	Yes	Yes
eTSEC59	False parity error at Tx startup	No plans to fix	Yes	Yes
eTSEC64	User-defined Tx preamble incompatible with Tx Checksum	No plans to fix	Yes	Yes
eTSEC67	ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software	No plans to fix	Yes	Yes
eTSEC68	Half-duplex collision on FCS of Short Frame may cause Tx lockup	Documentation update	Yes	Yes
eTSEC69	Ethernet controller does not exit from Magic Packet mode when an oddly formed Magic Packet is received	No plans to fix	Yes	Yes
eTSEC70	MAC: Malformed Magic Packet Triggers Magic Packet Exit	No plans to fix	Yes	Yes
eTSEC71	The value of TSEC_ID2 is incorrect	No plans to fix	Yes	Yes
eTSEC73	TxBD polling loop latency is 1024 bit-times instead of 512	No plans to fix	Yes	Yes
eTSEC74	MAC: Rx frames of length MAXFRM or MAXFRM-1 are marked as truncated	No plans to fix	Yes	Yes
eTSEC75	Misfiled Packets Due to Incorrect Rx Filer Set Mask Rollback	No plans to fix	Yes	Yes
eTSEC76	Excess delays when transmitting TOE=1 large frames	No plans to fix	Yes	Yes
eTSEC78	Controller may not be able to transmit pause frame during pause state	No plans to fix	Yes	Yes
eTSEC-A001	MAC: Pause time may be shorter than specified if transmit in progress	No plans to fix	Yes	Yes
A-006502	Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame	No plans to fix	Yes	Yes
IEEE 1588				
IEEE 1588_14	TxPAL timestamp uses TxBD snoop enable instead of Tx data	No plans to fix	Yes	Yes
IEEE 1588_16	Odd prescale values not supported	No plans to fix	Yes	Yes
IEEE 1588_17	Cannot use inverted 1588 reference clock when selecting eTSEC system clock as source	No plans to fix	Yes	Yes
IEEE 1588_19	Tx FIFO data parity error (DPE) may corrupt Tx timestamps if TMR_CTRL[TRTPE]=1	No plans to fix	Yes	Yes
IEEE 1588_20	eTSEC 1588: Write to reserved 1588 register space causes system hang	No plans to fix	Yes	Yes
IEEE1588-A001	Incorrect received timestamp or dropped packet when 1588 time-stamping is enabled	Fixed in Rev 1.1	Yes	No
A-007734	Stale time stamps and over-flow conditions can occur when using an external 1588 input clock at certain frequencies	No plans to fix	Yes	Yes
General				

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
General14	Electrostatic Discharge (ESD) may fail to meet the 2KV Human body body model (HBM)	No plans to fix	Yes	Yes
General16	Enabling I^2C could cause I^2C bus freeze when other I^2C devices communicate	No plans to fix	Yes	Yes
General17	DUART: Break detection triggered multiple times for a single break assertion	No plans to fix	Yes	Yes
PCIe				
PEX1	No support of PCI Express completions with BCM bit set (PCIX bridge interface)	No plans to fix	Yes	Yes
PEX2	DMA Interrupt descriptor race condition (IDRC)	No plans to fix	Yes	Yes
PEX5	PCI Express LTSSM may fail to properly train with a link partner following HRESET#	No plans to fix	Yes	Yes
PEX7	Recovery from hot reset or link down	No plans to fix	Yes	Yes
PCIe-A002	PCI Express Packets may be transmitted with excess data on x1 link	Partially fixed in 1.1	Yes	No
RTC				
A-003985	Real Time Counter Register (RTCTR) may not work correctly	Fixed in Rev 1.1	Yes	No
USB				
USB15	Read of PERIODICLISTBASE after successive writes may return a wrong value in host mode	No plans to fix	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
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
USB26	NackCnt field is not decremented when received NYET during FS/LS Bulk/Interrupt mode	No plans to fix	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
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
USB29	Priming ISO over SOF will cause transmitting bad packet with correct CRC	No plans to fix	Yes	Yes
USB31	Transmit data loss based on bus latency	No plans to fix	Yes	Yes
USB32	Missing SOFs and false babble error due to Rx FIFO overflow	No plans to fix	Yes	Yes
USB33	No error interrupt and no status will be generated due to ISO mult3 fulfillment error	No plans to fix	Yes	Yes
USB34	NAK counter decremented after receiving a NYET from device	No plans to fix	Yes	Yes
USB35	Core device fails when it receives two OUT transactions in a short time	No plans to fix	Yes	Yes
USB36	CRC not inverted when host under-runs on OUT transactions	No plans to fix	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
USB37	OTG Controller as Host does not support Data-line Pulsing Session Request Protocol	No plans to fix	Yes	Yes
USB-A001	Last read of the current dTD done after USB interrupt	No plans to fix	Yes	Yes
USB-A002	Device does not respond to INs after receiving corrupted handshake from previous IN transaction	No plans to fix	Yes	Yes
USB-A003	Illegal NOPID TX CMD issued by USB controller with ULPI interface	No plans to fix	Yes	Yes
USB-A005	ULPI Viewport not Working for Read or Write Commands With Extended Address	No plans to fix	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
A-003817	USB Controller locks after Test mode "Test_K" is completed	No plans to fix	Yes	Yes
A-003827	DATA PID error interrupt issued twice for the same high bandwidth ISO transfer	No plans to fix	Yes	Yes
A-003829	Host detects frame babble but does not halt the port or generate an interrupt	No plans to fix	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
A-003845	Frame scheduling robustness-Host may issue token too close to uframe boundary	No plans to fix	Yes	Yes

CPU6: DTLB LRU logic does not function correctly

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

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

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

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

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

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

Workaround: Use one of the following options:

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

Fix plan: No plans to fix

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

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

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

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

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

Sample Code:

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

Fix plan: No plans to fix

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

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

Impact: Disagreement with the JESD79-2C standard.

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

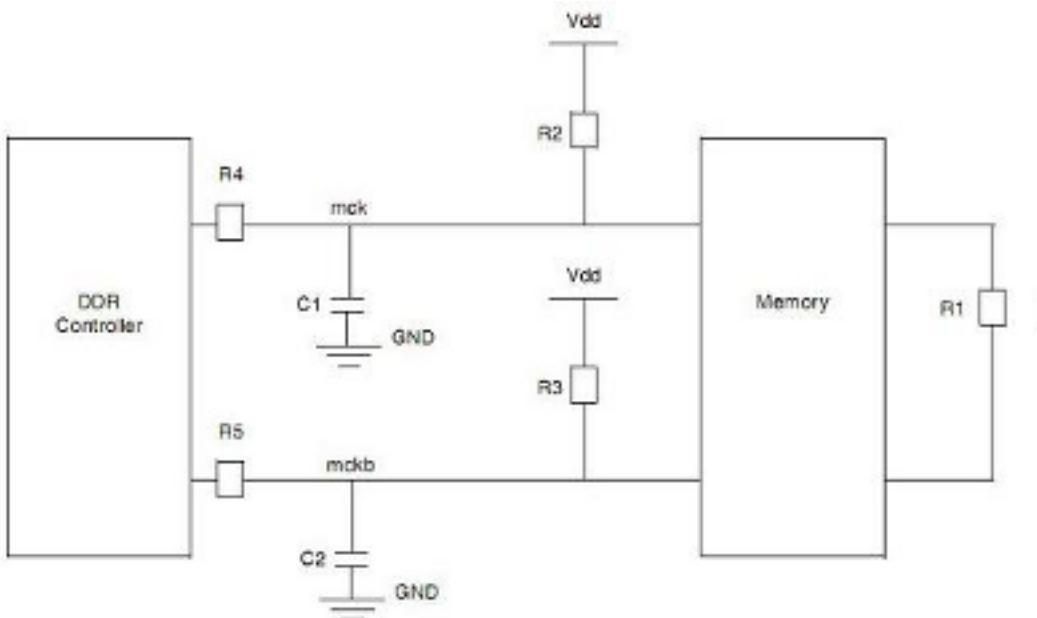

Figure 1. Recommended connections for DDR controller and memory

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

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

Component Placing:

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

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

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

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

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

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

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

Fix plan: No plans to fix

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

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

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

Workaround: None

Fix plan: No plans to fix

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

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

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

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

Fix plan: No plans to fix

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

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

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

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

Fix plan: No plans to fix

eSDHC16: Manual Asynchronous CMD12 abort operation causes protocol violations

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

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

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

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

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

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

Fix plan: No plans to fix

eSDHC23: CMD CRC error or CMD index error may be set for CMD without data while CMD with data is in progress

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

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

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

Fix plan: No plans to fix

eSDHC-A002: BLKATTR[BLKCNT] does not return to 0 after Transfer Complete for multi-block transfer

Description: For a multi-block transfer, when the XFERTYP[BCEN](block count enable) is 1, BLKATTR[BLKCNT] should decrement to 0 after Transfer Complete (TC)((IRQSTAT[TC] = 1). However, due to this erratum, BLKATTR[BLKCNT] returns to the initial value which was programmed while issuing the multi-block transfer command.

Impact: The value of BLKATTR[BLKCNT] is unreliable and cannot be read to confirm a Transfer Complete.

Workaround: For a multi-block transfer, after IRQSTAT[TC] = 1, software should not rely on the value of BLKATTR[BLKCNT].

Fix plan: No plans to fix

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

Affects: eSDHC

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

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

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

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

Fix plan: No plans to fix

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

Affects: eSDHC

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

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

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

Fix plan: No plans to fix

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

Affects: eSDHC

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

Impact: Causes system bus to hang.

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

Fix plan: No plans to fix

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

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

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

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

Fix plan: No plans to fix

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

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

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

Workaround: Use one of the following workarounds:

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

Fix plan: No plans to fix

eTSEC59: False parity error at Tx startup

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

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

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

eTSEC64: User-defined Tx preamble incompatible with Tx Checksum

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

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

Workaround: Use one of the following workarounds:

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

Fix plan: No plans to fix

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

eTSEC68: Half-duplex collision on FCS of Short Frame may cause Tx lockup

Description: In half-duplex mode, if a collision occurs in the FCS bytes of a short (fewer than 64 bytes) frame, then the Ethernet MAC may lock up and stop transmitting data or control frames. Only a reset of the controller can restore proper operation once it is locked up.

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

Workaround: **Option 1:**

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

Option 2:

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

Fix plan: Documentation update

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

eTSEC69: Ethernet controller does not exit from Magic Packet mode when an oddly formed Magic Packet is received

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

- A valid Ethernet header (Destination and Source Addresses)
- A valid FCS (CRC-32)
- A payload that includes the specific MagicPacket byte sequence at any offset from the start of data payload.

The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFs followed by 16 copies of the MAC's unique IEEE station address in the normal byte order for Ethernet addresses.

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

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

- FF_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

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

- FF_FF_FF_FF_FF_FF_01_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

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

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

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

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

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

Fix plan: No plans to fix

eTSEC70: MAC: Malformed Magic Packet Triggers Magic Packet Exit

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

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

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

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

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

Workaround: None

Fix plan: No plans to fix

eTSEC71: The value of TSEC_ID2 is incorrect

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

Impact: There is no functional impact.

Workaround: None

Fix plan: No plans to fix

eTSEC73: TxBD polling loop latency is 1024 bit-times instead of 512

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

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

Workaround: None

Fix plan: No plans to fix

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

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

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

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

Workaround: Option 1:

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

Option 2:

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

Fix plan: No plans to fix

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

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

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

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

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

Rule #2: <Mask = 0x0000_8000>

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

Rule #3: <Mask = 0x0000_8000>

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

Rule #4: <Mask = 0x0000_8000>

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

Rule #5: <Mask = 0x0000_0210>

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

Rule #6: <Mask = 0x0000_8000>

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

Rule #7: <Mask = 0xFF00_FFFF>

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

Rule #8: <Mask = 0xFF00_FFFF>

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

Rule #9: <Mask = 0xFF00_FFFF>

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

Rule #10: <Mask = 0xFF00_FFFF>

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

Rule #11: <Mask = 0xFF00_FFFF>

etc.

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

Rule #8: <Mask = 0xFF00_FFFF>

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

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

Rule #9: <Mask = 0x0000_8000>

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

Rule #10: <Mask = 0x0000_8000>

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

Rule #11: <Mask = 0x0000_8000>

Mask should be 0xFF00_FFFF.

etc.

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

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

Workaround: Use one of the following workarounds:

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

Fix plan: No plans to fix

eTSEC76: Excess delays when transmitting TOE=1 large frames

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

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

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

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

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

Fix plan: No plans to fix

eTSEC78: Controller may not be able to transmit pause frame during pause state

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

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

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

Workaround: None

Fix plan: No plans to fix

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

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

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

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

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

Fix plan: No plans to fix

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

Affects: eTSEC

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

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

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

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

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

Workaround: For GRS scenario:

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

MAX Rx reset procedure:

1) Clear MACCFG[RX_EN].

2) Wait three Rx clocks.

3) Set MACCFG2[RX_EN].

Fix plan: No plans to fix

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

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

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

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

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

Fix plan: No plans to fix

IEEE 1588_16: Odd prescale values not supported

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

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

Impact: Odd 1588 timer prescale values are not supported.

Workaround: Use only even timer prescale values.

Fix plan: No plans to fix

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

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

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

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

Workaround: None

Fix plan: No plans to fix

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

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

Example:

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

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

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

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

OR

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

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

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

Fix plan: No plans to fix

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

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

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

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

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

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

Fix plan: No plans to fix

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

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

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

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

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

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

Fix plan: Fixed in Rev 1.1

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

Affects: IEEE 1588

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

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

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

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

Workaround: Limit the 1588 input clock frequency to:

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

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

Fix plan: No plans to fix

General14: Electrostatic Discharge (ESD) may fail to meet the 2KV Human body body model (HBM)

Description: 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: No plans to fix

General16: Enabling I²C could cause I²C bus freeze when other I²C devices communicate

Description: When the I²C controller is enabled by software, if the signal SCL is high and the signal SDA is low, and the I²C address matches the data pattern on the SDA bus right after the enabling, an ACK is issued on the bus. The ACK is issued because the I²C controller detects a START condition due to the nature of the SCL and SDA signals at the point of enablement. When this occurs, it may cause the I²C bus to freeze. However, it happens very rarely due to the need of two conditions occurring at the same time.

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

Workaround: Use one of the following options:

- Enable the I²C controller before starting any I²C communications on the bus.

  This is preferred solution.

- If the I²C is configured as a slave, implement the following steps:

  a. Software enables the device by setting I2CnCR[MEN] = 1, and start a timer.
  b. Delay for four I²C bus clocks.
  c. Check Bus Busy bit (I2CnSR[MBB])

```
if MBB == 0
        Jump to Step 6; (Good condition. Go to normal operation.)
else
        Disable device (I2CnCR[MEN] = 0)
```

  d. Reconfigure all the I²C registers if necessary.
  e. Go back to Step 1
  f. Normal operation.

Fix plan: No plans to fix

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

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

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

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

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

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

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

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

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

This workaround applies to both polling and interrupt-driven implementations.

Fix plan: No plans to fix

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

Description: 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: 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: 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

PEX7: Recovery from hot reset or link down

Description: 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 MPC8308E 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.
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: 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: Partially fixed in 1.1

(CSB_CLK clock frequency should be greater than SD_REF_CLK clock frequency)

A-003985: Real Time Counter Register (RTCTR) may not work correctly

Affects: RTC

Description: When RTC_PIT_CLK is not connected, the RTCTR counter may not count correctly for both settings of the clock source: CSB clock source (RTCNR[CLIN]=0) or RTC_PIT_CLK clock source (RTCNR[CLIN]=1) for RTC.

Impact: RTC counter may not work correctly if RTC_PIT_CLK has no clock connected to it.

Workaround: Connect RTC_PIT_CLK pin to clock signal of any frequency.

Fix plan: Fixed in Rev 1.1

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

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

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

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

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

Fix plan: No plans to fix

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

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

Impact: None

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

Fix plan: No plans to fix

USB25: In host mode, when the software forces a port resume by writing into the FPR bit of the portsc register, the port change detect interrupt bit is falsely fired

Description: In host mode, a false "port change detect" interrupt is fired when the HCD (Host controller driver) resumes a suspended port by writing "1" to PORTSC[FPR] bit.

Impact: An interrupt is falsely fired when the software forces a port resume. There is an extra overhead to deal with the mis-fired interrupt.

Workaround: After setting PORTSC[FPR] and subsequent interrupt, the software should check the interrupt source, and clear USBSTS[PCI] bit, which corresponds to "port change detect" in Host mode.

Fix plan: No plans to fix

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

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

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

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

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

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

Fix plan: No plans to fix

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

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

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

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

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

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

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

Workaround: There is no workaround.

Fix plan: No plans to fix

USB31: Transmit data loss based on bus latency

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

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

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

Fix plan: No plans to fix

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

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

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

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

Fix plan: No plans to fix

USB34: NAK counter decremented after receiving a NYET from device

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

Impact: The NAK counter may be lower than expected.

Workaround: None

Fix plan: No plans to fix

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

Description: 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: 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: 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

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

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

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

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

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

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

Impact: There are two cases to consider:

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

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

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

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

Workaround:

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

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

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

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

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

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

Fix plan: No plans to fix

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

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

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

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

Workaround: None

Fix plan: No plans to fix

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

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

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

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

Workaround: None

Fix plan: No plans to fix

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

Affects: USB

Description: 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

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

Affects: USB

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

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

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

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

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

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

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

Workaround: None

Fix plan: No plans to fix

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

Affects: USB

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

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

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

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

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

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

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

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

Recovery:

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

Fix plan: No plans to fix

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

Affects: USB

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

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

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

Fix plan: No plans to fix

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

Affects: USB

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

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

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

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

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

Fix plan: No plans to fix

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

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

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

Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, Energy Efficient Solutions logo, Kinetis, mobileGT, PowerQUICC, Processor Expert, QorIQ, Qorivva, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, CoreNet, Flexis, Layerscape, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SafeAssure, SafeAssure logo, SMARTMOS, Tower, TurboLink, Vybrid, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2015 Freescale Semiconductor, Inc.