MPC8548E Chip Errata

This document details all known silicon errata for MPC8548E PowerQUICC III devices.

Table 1. Revision history

Revision	Date	Substantive changes
6	6/2012	• Renamed DUART 1 to A-004737 and CPU-A005 to A-001428. • Added CPU erratum A-005125. • In Table 2, updated the PVR value for 3.1.x from 0x8021_0023 to 0x8021_0022.
5	1/2012	• Added Rev 3.1.x silicon information • Added CPU erratum A-003477 • Removed eTSEC 102, fixed in Rev 2.0 silicon • Updated Fix plan for for CPU-A005, eTSEC 34, eTSEC 38, eTSEC 57, eTSEC 58, eTSEC 59, eTSEC 62, eTSEC 63, eTSEC 64, eTSEC 65, eTSEC 66, eTSEC 67,eTSEC68, eTSEC 69, eTSEC 71, eTSEC 72, eTSEC 73, eTSEC 74, eTSEC 76, eTSEC 77, eTSEC 78, eTSEC 79, eTSEC 81, eTSEC 82, eTSEC 83, eTSEC 84, eTSEC 85, eTSEC 87, eTSEC 88, eTSEC 89, eTSEC 90, eTSEC 92, eTSEC 93,eTSEC 94, eTSEC 95, eTSEC 98, eTSEC 99, eTSEC 101, eTSEC 104, eTSEC 105, eTSEC 106, eTSEC 107, eTSEC-A001, eTSEC-A004, PCI-Ex 35, PCI-Ex 37, PCI-Ex 38, PCI-Ex 39, PCI-Ex 42, PCIe-A001, SRIO-A002, GEN 11, GEN13, I2C 2, DMA 1, and DMA 2 • In Table 2, Corrected SVR values for 'without security.' The leading digit was missing. • Modified CPU 4 workaround • In Table 3, corrected silicon revisions impacted for JTAG 2 • Modified eTSEC 93 description
4	4/2011	• Added eTSEC-A005 • Modified impact for CPU 3 and SEC-A001

Table continues on the next page...

Table 1. Revision history (continued)

Revision	Date	Substantive changes
3	10/2010	• Added CPU-A001, CPU-A005, eTSEC 106, eTSEC 107, eTSEC-A001, eTSEC-A002, eTSEC-A004, PCI-Ex 42, PCIe-A001, SEC-A001, SRIO-A002, SRIO-A004, I2C 2 and I2C 3. • Added OMnDATR note to Description on SRIO42. • Removed PCI-Ex 40 due to duplicate entry. • Updated disposition for eTSEC 104, SEC 5, SEC 6. • Updated workaround for eTSEC 79. • In Table 3, removed Rev. 1.1, 1.1.1, and 1.1.2 columns. • In DUART 1, removed stray text after fixplan, incorrectly showing in Rev. 2. • In SEC 8, changed byte to bit throughout. • In PM2, changed PMCx value in workaround. • Removed PM3, it was fixed in Rev. 2.0. • In eTSEC72, changed “Compound Rule Example” table entry 2, RQPROP value in workaround.
2	02/2009	• Changed DDR 20 by adding note explaining location of debug registers used in work around. • Added SRIO 41. • Added SerDes 1. • Added eTSEC 105. • Modified disposition for eTSEC 39. • Modified workaround for eTSEC 79. • Removed eTSEC 103. • Added SRIO 42. • Updated information in Table 2, “Revision Level to Part Marking Cross-Reference.”
1	01/2009	• Added PCI-Ex 41 and SEC 8. • Changed 5M to 6M for supported mask set in Table 2.
0	12/2008	Initial release

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

Table 2. Revision Level to Part Marking Cross-Reference

Revision	e500v2 Core Revision	Processor Version Register Value	System Version Register Value	Device Marking
2.0.1	2.0	0x8021_0020	With security 0x8039_0020	2M39E
			Without security 0x8031_0020	3M39E
2.1.1	2.2	0x8021_0022	With security 0x8039_0021 Without security 0x8031_0021	6M39E
2.1.2	2.2	0x8021_0022	With security 0x8039_0021 Without security 0x8031_0021	7M39E
3.1.x	2.3	0x8021_0022	With security 0x8039_0031 Without security 0x8031_0031	1N06D, 2N06d

The following table summarizes all known errata on the MPC8548E device and lists the corresponding silicon revision level to which it applies. A ‘Yes’ entry indicates the erratum applies to a particular revision level, while a 'No' entry means it does not apply.

Table 3. Summary of Silicon Errata and Applicable Revision

Errata	Name	Projected Solution	Silicon Rev.		
			2.0	2.1.x	3.1.x
CPU					
CPU 2	A core hang possible while executing a msync or mbar 0 instruction and a snoopable transaction from an I/O master tagged to make quick forward progress is present.	Fixed in Rev 2.1	Yes	No	No
CPU 3	"mbar MO = 1" instruction fails to order caching-inhibited guarded loads and stores	No plans to fix	Yes	Yes	Yes
CPU 4	Single-precision floating-point zero value may have the wrong sign	No plans to fix	Yes	Yes	Yes
CPU 5	Branch Fails to Decrement CTR in Presence of D-cache Parity Error	No plans to fix	Yes	Yes	Yes
CPU 6	Incorrect Value May be Loaded Into SRR0 on ITLB Miss When Executing **icbtls**	No plans to fix	Yes	Yes	Yes
CPU-A001	Flash-Invalidation of TLB1 with "mtspr MMUCSR0" may fail	No plans to fix	Yes	Yes	Yes
A-001428(CPU-A005)	Enabling IEEE 754 exceptions can cause errors	Fixed in Rev 3.1.x	Yes	Yes	No
A-003477	Phantom branches in the BTB may not be correctly invalidated	No plans to fix	Yes	Yes	Yes
A-005125	In a very rare condition, a system hang is possible when the e500 core initiates a guarded load to PCI/PCIe/sRIO while the PCI/PCIe/sRIO performs a coherent write to memory.	No plans to fix	Yes	Yes	Yes
eTSEC					
eTSEC 29	Frame is dropped with collision and HALFDUP[Excess Defer] = 0	No plans to fix	Yes	Yes	Yes
eTSEC 34	Transmit frames aborted under 16-bit FIFO GMII-style mode	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 36	TX_EN of eTSEC1 and eTSEC2 may be driven randomly during reset	Fixed in Rev 2.1	Yes	No	No
eTSEC 37	eTSEC Parser does not properly parse L3 fields	Fixed in Rev 2.1	Yes	No	No
eTSEC 38	WWR bit Anomaly	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 39	Parsing of tunneled IP packets not supported	No plans to fix	Yes	Yes	Yes
eTSEC 40	Magic packet does not wake device from a SLEEP state	No plans to fix	Yes	Yes	Yes
eTSEC 44	FIFO8, FIFO16 TX hang	Fixed in Rev 2.1	Yes	No	No
eTSEC 45	Tx data corruption in FIFO16 mode	Fixed in Rev 2.1	Yes	No	No
eTSEC 46	RSTAT[RXF0] set regardless of destination ring if WWR=0	Fixed in Rev 2.1	Yes	No	No
eTSEC 47	VLAN strip must be disabled if the parser is disabled	Documentation update	Yes	No	No
eTSEC 48	Receive frame filer must be disabled if the parser is disabled	Documentation update	Yes	No	No

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			2.0	2.1.x	3.1.x
eTSEC 49	Tx IP and TCP/UDP Checksum Generation not supported for some Tx FCB offsets	Fixed in Rev 2.1	Yes	No	No
eTSEC 50	Value of TBYT register may not match bytes transmitted for truncated frames	Documentation update	Yes	No	No
eTSEC 51	Missing basic integrity check for parsing Tunneled IP packets	No plans to fix	Yes	Yes	Yes
eTSEC 52	Error in arbitrary extraction offset	Fixed in Rev 2.1	Yes	No	No
eTSEC 55	Transmit jumbo frames greater than 2400 bytes may cause lost data, loss of BD synchronization, or false underrun error	Fixed in Rev 2.1	Yes	No	No
eTSEC 56	Parser results may be lost if TCP/UDP checksum checking is enabled	Fixed in Rev 2.1	Yes	No	No
eTSEC 57	RQUEUE[EXn] bits have no effect	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 58	Parsing of MPLS label stack or non-IPv4/IPv6 label not supported	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 59	Arbitrary extraction on short frames uses data from previous frame	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 60	Some combinations of Tx packets may trigger a false Data Parity Error (DPE)	Fixed in Rev 2.1	Yes	No	No
eTSEC 61	eTSEC Data Parity Error (DPE) does not abort transmit frames	Partial fix in Rev 2.1	Yes	Yes	Yes
eTSEC 62	eTSEC half duplex receiver packet corruption	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 63	May drop Rx packets in non-FIFO modes with lossless flow control enabled	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 64	Back-to-back IPv6 routing headers not supported by parser	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 65	RxBD[TR] not asserted during truncation when last 4 bytes match CRC	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 66	Parser continues parsing L4 fields when RCTRL[PRSDEP] set for L2 and L3 fields only	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 67	Fetches with errors not flagged, may cause livelock or false halt	Partially fixed in Rev 3.1.x	Yes	Yes	Yes
eTSEC 68	Rx TCP/UDP checksum checking may be incorrect while operating at low frequencies in FIFO mode	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 69	Filer does not support matching against broadcast address flag PID1[EBC]	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 70	eTSEC does not support parsing of LLC/SNAP/VLAN packets	No plans to fix	Yes	Yes	Yes
eTSEC 71	eTSEC filer reports incorrect Ether-types with certain MPLS frames	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 72	Compound filer rules do not roll back the mask	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 73	Incomplete frame with error causes false CR error on next frame	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 74	Parser does not check VER/TYPE of PPPoE packets	Fixed in Rev 3.1.x	Yes	Yes	No

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			2.0	2.1.x	3.1.x
eTSEC 75	Back-to-back Rx frames may lose parser results of second frame	Fixed in Rev 2.1	Yes	No	No
eTSEC 76	RMCA, RBCA counters do not correctly count valid VLAN tagged frames	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 77	Tx errors truncate packets without error in 8-bit Encoded FIFO mode	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 78	No parser error for packets containing invalid IPv6 routing header packet	Partially fixed in Rev 3.1.x	Yes	Yes	Yes
eTSEC 79	Generation of Ethernet pause frames may cause Tx lockup and false BD close	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 80	eTSEC parser does not perform length integrity checks	No plans to fix	Yes	Yes	Yes
eTSEC 81	eTSEC does not verify IPv6 routing header type field	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 82	Transmission of truncated frames may cause hang or lost data	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 83	L3 fragment frame files on non-existent source/destination ports	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 84	Multiple BD frame may cause hang	Fixed in Rev 3.1.x if multiple Tx queues enabled.	Yes	Yes	Yes
eTSEC 85	TxBD[TC] is not reliable in 16-bit FIFO modes	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 86	eTSEC receivers may not be properly initialized	Fixed in Rev 2.1	Yes	No	No
eTSEC 87	Arbitrary Extraction cannot extract last data bytes of frame	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 88	False TCP/UDP and IP checksum error in FIFO mode without CRC appending	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 89	Frames greater than 9600 bytes with TOE = 1 will hang controller	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 90	Setting RCTRL[LFC] = 0 may not immediately disable LFC	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 91	VLAN Insertion corrupts frame if user-defined Tx preamble enabled	No plans to fix	Yes	Yes	Yes
eTSEC 92	False parity error at Tx startup	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 93	Transmit fails to utilize 100% of line bandwidth	Partially fixed in Rev 3.1.x	Yes	Yes	Yes
eTSEC 94	Rx packet padding limitations at low clock ratios	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 95	False TCP/UDP checksum error for some values of pseudo header Source Address	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 96	Unexpected babbling receive error in FIFO modes	No plans to fix	Yes	Yes	Yes
eTSEC 97	User-defined Tx preamble incompatible with Tx Checksum	No plans to fix	Yes	Yes	Yes
eTSEC 98	FIFO16 interface encoded mode maximum frequency is 1/4.2 platform clock	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 99	ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 100	Half-duplex collision on FCS of Short Frame may cause Tx lockup	Documentation update	Yes	Yes	Yes

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			2.0	2.1.x	3.1.x
eTSEC 101	Magic Packet Sequence Embedded in Partial Sequence Not Recognized	Partially fixed in Rev 3.1.x	Yes	Yes	Yes
eTSEC 104	Rx may hang if RxFIFO overflows	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 105	MAC: Malformed Magic Packet Triggers Magic Packet Exit	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 106	Excess delays when transmitting TOE=1 large frames	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC 107	Controller may not be able to transmit pause frame during pause state	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC-A001	MAC: Pause time may be shorter than specified if transmit in progress	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC-A002	Incomplete GRS or invalid parser state after receiving a 1- or 2-byte frame	No plans to fix	Yes	Yes	Yes
eTSEC-A004	User-defined preamble not supported at low clock ratios	Fixed in Rev 3.1.x	Yes	Yes	No
eTSEC-A005	Rx may hang on eTSEC3 and eTSEC4 under extreme temperature, low Vdd and heavy traffic	Fixed in Rev 2.1.3	Yes	No	No
PEX					
PCI-Ex 35	Rx detection by transmitter does not pass high receiver impedance test	Fixed in Rev 3.1.x	Yes	Yes	No
PCI-Ex 37	Completion Timeout error disable corrupts CRS threshold error data	Fixed in Rev 3.1.x	Yes	Yes	No
PCI-Ex 38	PCI Express LTSSM may fail to properly train with a link partner following $\overline{\text{HRESET}}$	Fixed in Rev 3.1.x	Yes	Yes	No
PCI-Ex 39	No mechanism for recovery from hang after access to down link	Fixed in Rev 3.1.x	Yes	Yes	No
PCI-Ex 41	PCI Express x8 mode requires minimum platform frequency of 527 MHz	No plans to fix	Yes	Yes	Yes
PCI-Ex 42	Reads to PCI Express CCSRs or local config space temporarily return all Fs	Fixed in Rev 3.1.x	Yes	Yes	No
PCIe-A001	PCI Express Hot Reset event may cause data corruption	Fixed in Rev 3.1.x	Yes	Yes	No
SEC					
SEC 5	AES-CTR mode data size error	Fixed in Rev 2.1	Yes	No	No
SEC 6	Single descriptor SRTP error	Fixed in Rev 2.1	Yes	No	No
SEC 7	AES-CCM ICV checking write-back error	No plans to fix	Yes	Yes	Yes
SEC 8	Kasumi hardware ICV checking does not work	No plans to fix	Yes	Yes	Yes
SEC-A001	Channel Hang with Zero Length Data	No plans to fix	Yes	Yes	Yes
SRIO					
SRIO 39	Serial RapidIO atomic operation erratum	No plans to fix	Yes	Yes	Yes
SRIO 41	Serial RapidIO Packets with errors are not ignored by the controller while in input-retry-stopped state	No plans to fix	Yes	Yes	Yes
SRIO 42	Message unit cannot generate messages with priority 0	No plans to fix	Yes	Yes	Yes
SRIO-A002	SRIO reset command does not result in device reset	Fixed in Rev 3.1.x	Yes	Yes	No
SRIO-A004	SRIO controller may incorrectly transmit or block responses when PmCCSR[OPE] = 0	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			2.0	2.1.x	3.1.x
	DDR				
DDR 13	Memory contents may not be retained during $\overline{\text{HRESET}}$ sequence	No plans to fix	Yes	Yes	Yes
DDR 14	The automatic CAS-to-Preamble feature of the DDR controller can calibrate to incorrect values	No plans to fix	Yes	Yes	Yes
DDR 15	Automatic calibration hardware may calibrate to an invalid driver impedance	Fixed in Rev 2.1	Yes	No	No
DDR 16	On-die termination at the DDR IOs has been measured 75 Ω too high	Fixed in Rev 2.1	Yes	No	No
DDR 17	DDR performance monitoring and tracing functionality does not work	Fixed in Rev 2.1	Yes	No	No
DDR 18	Automatic data initialization and DLL resets to DRAM are not performed correctly if CS2 and CS3 are interleaved	No plans to fix	Yes	Yes	Yes
DDR 19	DDR IOs default receiver biasing may not work across voltage and temperature	Fixed in Rev 2.1	Yes	No	No
DDR 20	MCKE signal may not function correctly at assertion of $\overline{\text{HRESET}}$	No plans to fix	Yes	Yes	Yes
	GEN				
GEN 10	Device may fail DDR pins during IEEE Std 1149.1™ EXTEST mode	No plans to fix	Yes	Yes	Yes
GEN 11	Some pins do not meet 500V CDM ESD criteria	Fixed in Rev 3.1.x	Yes	Yes	No
GEN 13	CPU write to platform failures in all core, platform, and SYSCLK frequency combinations	Fixed in Rev 3.1.x	Yes	Yes	No
GEN 14	CPU-only reset may result in a failure in the platform logic	No plans to fix	Yes	Yes	Yes
	SerDes				
SERDES 1	SerDes lane power down function may cause training to fail.	No plans to fix	Yes	Yes	Yes
	LBC				
LBIU 3	LBIU Transactions when clock divider LCRR[CLKDIV] is changing	No plans to fix	Yes	Yes	Yes
LBIU 4	LGTA/LUPWAIT assertion in PLL-bypass mode misrepresented	No plans to fix	Yes	Yes	Yes
LBIU 5	UPM does not have indication of completion of a RUN PATTERN special operation	No plans to fix	Yes	Yes	Yes
	PIC				
PIC 4	PIC soft reset not clearing MSIRn registers correctly	No plans to fix	Yes	Yes	Yes
PIC 5	MSIMR De-featured	Documentation update	Yes	Yes	Yes
PIC 6	MER, Interrupt will not be forwarded to destination	No plans to fix	Yes	Yes	Yes
PIC 7	PCI Express MSI other than interrupt 0 not supported via hardware	No plans to fix	Yes	Yes	Yes
	PMON				
PM 1	Some local bus events are not counted correctly in the performance monitor	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			2.0	2.1.x	3.1.x
PM 2	A performance monitor time base transition event will not freeze the performance monitor counters and may not cause an exception	No plans to fix	Yes	Yes	Yes
	I2C				
I2C 2	Enabling I^2C could cause I^2C bus freeze when other I^2C devices communicate	Fixed in Rev 3.1.x	Yes	Yes	No
I2C 3	I2C boot sequencer cannot issue snoopable writes	Fixed on Rev 2.1	Yes	No	No
	DMA				
DMA 1	$\overline{\text{DMA\_DACK}}$ bus timing violation when operating in external DMA master mode	Partially fixed in Rev 3.1.x	Yes	Yes	Yes
DMA 2	Transfer error reported for wrong channel	Fixed in Rev 3.1.x	Yes	Yes	No
	DUART				
A-004737	BREAK detection triggered multiple times for a single break assertion	No plans to fix	Yes	Yes	Yes
	PCI				
PCI 4	PCI-X inbound reads resulting in a split completion response can cause a corrupted response	No plans to fix	Yes	Yes	Yes
PCI 5	Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang	No plans to fix	Yes	Yes	Yes
PCI 6	PCI/PCI-X erroneous error detection	Fixed in Rev 2.1	Yes	No	No
PCI 7	Asynchronous mode PCI1 and PCI2 input hold violation	Fixed in Rev 2.1	Yes	No	No
PCI 8	Device reports wrong PCI Vendor ID	No plans to fix	Yes	Yes	Yes
PCI 9	Master-Abort issued incorrectly for outbound DAC with subtractive decode	No plans to fix	Yes	Yes	Yes
	JTAG				
JTAG 2	TMS requires hold time beyond the fall of TCK	Fixed in Rev 2.1	Yes	No	No

CPU 2: A core hang possible while executing a msync or mbar 0 instruction and a snoopable transaction from an I/O master tagged to make quick forward progress is present.

Description: Transactions from I/O masters (eTSEC, Security, DMA, PCI, PCI-X, SRIO, PCI-Exp, HT) are sometimes elevated in priority in the Coherency Module. If a **msync** or **mbar** 0 instruction is being executed inside of the core while one of these high priority transactions from I/O is “snooped” to the core's cache, a hang can result.

Impact: The core hangs and all I/O traffic going through the Coherency Module comes to a halt. The application must be using msync or mbar 0 instructions inside of the core for this problem to appear. Note also that ATMU and or setting of the buffer descriptors must be setup to permit I/O masters to perform coherent transfers for such a core hang to occur.

Note that peer-to-peer type traffic across the OCN complex continues without lockup.

Workaround: Setting the Coherency Module debug register at CCSR offset 0x0_1010, bit 15 to a logic 1 prevents this type of lockup. This disables the state machine inside of the Coherency Module that elevates these transactions from I/O masters to highest priority. The result is that all transactions that the core initiates are always permitted to compete on an equal basis on the Core Connect Bus (CCB) so that the core will not hang.

Fix plan: Fixed in Rev 2.1

CPU 3: "mbar MO = 1" instruction fails to order caching-inhibited guarded loads and stores

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

Assume the following instruction sequence:

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

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

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

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

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

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

Workaround: Use one of the following options:

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

The **msync** instruction can also be used as barrier, independent of the memory control attributes (for example, presence or absence of the caching inhibited attribute).

Fix plan: No plans to fix

CPU 4: Single-precision floating-point zero value may have the wrong sign

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

CPU 5: Branch Fails to Decrement CTR in Presence of D-cache Parity Error

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

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

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

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

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

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

Workaround: None

Fix plan: No plans to fix

CPU 6: Incorrect Value May be Loaded Into SRR0 on ITLB Miss When Executing icbtls

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

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

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

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

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

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

Fix plan: No plans to fix

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

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

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

Workaround: Use one of the following options:

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

Fix plan: No plans to fix

A-001428: Enabling IEEE 754 exceptions can cause errors

Affects: CPU

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

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

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

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

Workaround: Use one of the following options:

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

Fix plan: Fixed in Rev 3.1.x

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

Affects: CPU

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

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

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

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

Impact: Performance may be impacted. This errata could occur when switching across processes that share the same effective address space.

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

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

Each workaround impacts performance depending on the application.

Fix plan: No plans to fix

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

Affects: CPU

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

Impact: When no further forward progress is made to and from the e500 coherency module (ECM), the system may hang. If set, the watch dog timer then triggers the system to reboot.

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

Fix plan: No plans to fix

eTSEC 29: Frame is dropped with collision and HALFDUP[Excess Defer] = 0

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

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

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

Fix plan: No plans to fix

eTSEC 34: Transmit frames aborted under 16-bit FIFO GMII-style mode

Description: If the eTSEC is connected via a 16-bit FIFO packet interface with GMII signaling, and the CCB (platform) clock to Tx clock ratio is less than 4.2:1, transmit packets may be lost due to underrun. For example, for CCB (platform) clock of 533 MHz, the maximum speed of the Tx clock is 125 MHz. Also the minimum Inter Packet Gap (IPG) between packets at this ratio is 8 cycles. The logic in the transmit synchronizer needs the extra timing margin to recover between data beats when in 16-bit FIFO mode. This limitation is restricted to the FIFO16 GMII transmit path only. The receive path will operate with the CCB clock: Tx clock ratios as low as 3.2:1. FIFO8 and FIFO16 encoded modes can also operate at 3.2:1.

Impact: FIFO16 GMII is limited to a maximum transmit clock frequency of 125 MHz with platform clock at 533 MHz and a maximum transmit clock frequency of 155 MHz with platform clock at 667 MHz in Rev 2.0 silicon.

Workaround: Use of encoded mode for any 16 bit FIFO packet interface instead of GMII style signaling is recommended. The FIFO Encoded mode will not abort packets if underrun occurs; instead, it will assert idle bytes during the data stream.

Fix plan: Fixed in Rev 3.1.x

eTSEC 36: TX_EN of eTSEC1 and eTSEC2 may be driven randomly during reset

Description: During POR, pin TX_EN of eTSEC1 & eTSEC2 may be driven randomly for a period before being driven LOW while eTSEC3 & eTSEC4 tri-state this pin during the whole period where HRESET is asserted.

Impact: The PHY or external ASIC/FPGA would see the false active TX_EN during reset when connected to eTSEC1 and eTSEC2.

Workaround: To prevent unexpected behavior from the external devices (PHY or ASIC/FPGA) due to invalid TX_EN, software can reset the external devices after resetting the MPC8548E. This is required only if the external devices use a different reset source than one for MPC8548E.

Fix plan: Fixed in Rev 2.1

All TX_EN pins will be tri-stated during HRESET. assertion.

eTSEC 37: eTSEC Parser does not properly parse L3 fields

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

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

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

Fix plan: Fixed in Rev 2.1

eTSEC 38: WWR bit Anomaly

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

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

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 39: Parsing of tunneled IP packets not supported

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

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

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

Impact: Validly encapsulated tunneled IP packets may cause a false parser error or false TCP/UDP checksum error.

Malformed tunneled packets may be received without a parser error.

Tunneled packets with an actual TCP/UDP checksum error may fail to report a checksum error.

Workaround: If L3 or L4 parsing is enabled and tunneled packets are expected, software must examine each packet header to see if it is a tunneled IP packet. If the packet is a tunneled IP packet, software should ignore any parser or checksum error.

Fix plan: No plans to fix

Revision 2.1.x will continue to not support parsing of tunneled IP packets. However, false checksum and parser errors on tunneled IP packets will not occur. The inner header of those packets will not be parsed. The FCB will contain the correct parser information for the outer header and the next header field will correctly indicate the tunneled type. If tunneled packets are expected, software will need to check the FCB and flag any IP packets with the next header field = IP for further processing. Software will no longer need to check every packet.

eTSEC 40: Magic packet does not wake device from a SLEEP state

Description: There is a problem waking the device from a SLEEP state with a magic packet. When a magic packet is received on an eTSEC which is configured to come out of a low-power state (DOZE, NAP, SLEEP), the device is supposed to generate an interrupt to the interrupt controller which in turn gets the chip out of the low power state. Current versions of the device perform the correct action for DOZE and NAP, but not for SLEEP.

Impact: Magic packet cannot be used to get the device out of a SLEEP state.

Workaround: There is no work around for this erratum. Use DOZE or NAP low-power states for applications that use the magic packet to get the device out of a low-power state.

Fix plan: No plans to fix

eTSEC 44: FIFO8, FIFO16 TX hang

Description: The transmit state machine can hang in FIFO8 or FIFO16 mode.

Impact: Both encoded- and GMII-type FIFO8 and FIFO16 modes are non-functional in silicon revision 2.0.

Workaround: None.

Fix plan: Fixed in Rev 2.1

eTSEC 45: Tx data corruption in FIFO16 mode

Description: The eTSEC may occasionally corrupt data on transmit in FIFO16 encoded mode when TX flow control is enabled.

Impact: The problem only occurs when the platform is running at less than 4.2x the GTX_CLK in silicon revision 2.0 devices.

Workaround: Run the FIFO interface slower than 1/4.2 the platform frequency or run in the configuration Tx flow control disabled (FIFOCFG[TFC]=0). The eTSEC may experience underruns when running at less than 1/4.2, and TFC=0.

Fix plan: Fixed in Rev 2.1

eTSEC 46: RSTAT[RXF0] set regardless of destination ring if WWR=0

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

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

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

See also eTSEC 38 (WWR Bit Anomaly) for a description of other software requirements when WWR=0.

Fix plan: Fixed in Rev 2.1

eTSEC 47: VLAN strip must be disabled if the parser is disabled

Description: The current documentation says that VLAN tags are deleted only if RCTRL[VLEX]=1 and RCTRL[PRSDEP] != 0. In fact, the VLAN tag is deleted regardless of the setting of RCTRL[PRSDEP]. Since there is no way to recover the deleted VLAN tag if the parser is disabled, RCTL[VLEX] should never be set to 1 if RCTRL[PRSDEP]=0.

Impact: VLAN tags may be deleted if parser is disabled under illegal configuration setting.

Workaround: Always set RCTRL[VLEX]=0 if RCTRL[PRSDEP]=0.

Fix plan: Documentation update

eTSEC 48: Receive frame filer must be disabled if the parser is disabled

Description: The current documentation states that receive frame filing is enabled only if RCTRL[FILREN]=1 and RCTRL[PRSDEP] != 0. This is not true. The receive frame filer is enabled regardless of the setting of RCTRL[PRSDEP]. Since the filer is not intended to function if the parser is disabled, RCTL[FILREN] should never be set to 1 if RCTRL[PRSDEP]=0.

Impact: Improper filer behavior.

Workaround: User code must clear RCTRL[FILREN]=0 if RCTRL[PRSDEP]=0.

Fix plan: Documentation update

eTSEC 49: Tx IP and TCP/UDP Checksum Generation not supported for some Tx FCB offsets

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

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

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

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

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

Fix plan: Fixed in Rev 2.1

eTSEC 50: Value of TBYT register may not match bytes transmitted for truncated frames

Description: The value of the TBYT register may not match the actual number of bytes transmitted if the frame is truncated because it exceeds MAXFRM.

Impact: TBYT may be incremented for several bytes after MAXFRM limit is exceeded. The amount TBYT is incremented for a frame is guaranteed to be less than or equal to the original frame size.

Workaround: Do not transmit data larger than MAXFRM size.

Fix plan: Documentation update

Update documentation to indicate TBYT may be larger than the actual number of bytes transmitted if a frame is truncated because it exceeds MAXFRM.

eTSEC 51: Missing basic integrity check for parsing Tunneled IP packets

Description: eTSEC verifies basic integrity in outer IP headers, but not inner ones. It is good practice to verify packet header integrity on fields that are defined to contain certain values.

Two basic integrity checks involve the following:

- Ensuring that the version field of the IP header is a 4 or a 6, and corresponds to the previous headers encoding. Else, this is a parse error. For non-tunneled packets eTSEC perform this functionality properly.
- In the case of IPv4, the minimum header length allowed by the standard is 20 bytes, encoded as 0x5 in the header length field. Values less that 5 should be considered parse error. For non-tunneled packets eTSEC perform this functionality properly.

Impact: eTSEC will parse and file these irregular packets as valid encodings.

Workaround: If L3 or L4 parsing is enabled and tunneled packets are expected, software should perform basic checking on receive packets to see if they are tunneled IP packets. If so, the software should perform these checks.

Fix plan: No plans to fix

eTSEC 52: Error in arbitrary extraction offset

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

Impact: The following bytes cannot be extracted:

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

Note that PPOE and SNAP count as two tags each.

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

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

With one tag (VLAN/MPLS/PPOE):

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

With 2 tags (VLAN/MPLS/PPOE):

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

Fix plan: Fixed in Rev 2.1

eTSEC 55: Transmit jumbo frames greater than 2400 bytes may cause lost data, loss of BD synchronization, or false underrun error

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

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

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

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

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

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

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

Fix plan: Fixed in Rev 2.1

eTSEC 56: Parser results may be lost if TCP/UDP checksum checking is enabled

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

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

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

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

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

Fix plan: Fixed in Rev 2.1

eTSEC 57: RQUEUE[EXn] bits have no effect

Description: The RQUEUE[EXn] bits are intended to allow ring-by-ring control of stashing (allocate into L2 cache) function. The EXn bits have no effect in the eTSEC IP, so stashing is completely controlled by the settings of the DMA ATTR register.

Impact: Stashing cannot be disabled per ring. Stashing is a performance hint to the L2 cache to allocate data expected to be accessed by the CPU. If stashing is unsuccessful or disabled, the data is fetched from main memory.

Workaround: None

Fix plan: Fixed in Rev 3.1.x

eTSEC 58: Parsing of MPLS label stack or non-IPv4/IPv6 label not supported

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 59: Arbitrary extraction on short frames uses data from previous frame

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

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 60: Some combinations of Tx packets may trigger a false Data Parity Error (DPE)

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

The packet combinations that trigger the error are as follows:

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

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

Workaround: Although it is possible to prevent false parity error indications by disabling SRAM parity detection (accomplished by setting EDIS[DPEDIS] = 1), this can have the undesirable effect of masking indications of real parity errors. Unless the specific application is experiencing a significant number of false parity errors that are resulting in unsatisfactory performance degradation, it is recommended that SRAM parity detection remain enabled. Please refer to eTSEC 61 for more information.

Fix plan: Fixed in Rev 2.1

eTSEC 61: eTSEC Data Parity Error (DPE) does not abort transmit frames

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

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

In 8-bit and 16-bit GMII-style packet FIFO mode the TSECn_TX_ER signal should be asserted to indicate there was an error. However, due to this erratum a parity error will not cause the eTSEC to assert this signal.

In 8-bit encoded packet FIFO mode, due to the limited number of pins, the device is not designed to signal a packet was sent with errors. Therefore, this erratum does not apply when operating in this mode.

In 16-bit encoded packet FIFO mode TXC[2:0] = 011 indicates a packet was sent with an error, where TXC[2:0] = {TX_ENn+1, TX_ERn, TX_ENn}. However, due to this erratum a parity error will not set TXC[2:0] = 011 to indicate an error.

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

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

Workaround: None

Fix plan: Partial fix in Rev 2.1

Silicon revision 2.1 performs as follows:

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

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

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

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

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

   In MAC modes:

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

   In FIFO modes:

   a. Clear FIFOCFG[TXE]
   b. Set FIFOCFG[TXE]
5. Clear all TSTAT[THLTn] bits

eTSEC 62: eTSEC half duplex receiver packet corruption

Description: When eTSEC is configured to run in half-duplex configuration, it relies on the PHY to isolate its receiver from receive data during packet transmission. If the eTSEC receives data (RX_DV=1) on a port while eTSEC is simultaneously transmitting (TX_EN=1) on that same port, it will receive the corrupted packet data. If the receive port is operating in promiscuous mode where no frame filtering is done at the MAC, corrupted packet data may be written to system memory.

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

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

Workaround: The eTSEC can be configured through either the MAC address filter or the filer to discard such packets that are received in this manner through the use of MAC addresses filtering match and drop. The receive MIB counters may still incorrectly count packet errors.

Use eTSEC loopback mode configuration for full-duplex operation.

Fix plan: Fixed in Rev 3.1.x

eTSEC 63: May drop Rx packets in non-FIFO modes with lossless flow control enabled

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

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

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

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

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

Workaround:

- **Option 1:**

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

- **Option 2:**

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

NOTE

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 64: Back-to-back IPv6 routing headers not supported by parser

Description: Upon encountering back to back IPv6 routing extension headers following an IPv6 header, the eTSEC stops further parsing, and place 0xFF in the RxFCB[PRO], and RQFPR,pid = 0xB[L4P]. If there are 2 or more IPv6 routing extension headers, and there is either an IPv6 hop-by-hop extension header (see reference to RFC2460 below) or an IPv6 destination options header or both between the routing headers, then the eTSEC continues to parse the sequence.

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 65: RxBD[TR] not asserted during truncation when last 4 bytes match CRC

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

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

Workaround: None

Fix plan: Fixed in Rev 3.1.x

eTSEC 66: Parser continues parsing L4 fields when RCTRL[PRSDEP] set for L2 and L3 fields only

Description: RCTRL[PRSDEP] has encodings for the following:

- Disabling the parser (b00)
- Enabling parsing for L2 fields only (b01)
- Enabling parsing for L2 and L3 fields only (b10)
- Enabling parsing for L2, L3 and L4 fields (b11)

In the case of setting RCTRL[PRSDEP] = b10, the eTSEC does not stop its parsing activity after the L3 fields have been identified. Instead, it continues to parse the L4 fields, attempting to identify any supported L4 protocols and updating the RxFCB and filer PID = 1 fields TCP and UDP.

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 67: Fetches with errors not flagged, may cause livelock or false halt

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

Scenario 1

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

Scenario 2

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

Scenario 3

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

Scenario 4

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

Scenario 5

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

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

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

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

Workaround: All scenarios:

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

Transmit buffer descriptor work around:

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

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

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

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

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

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

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

Fix plan: Partially fixed in Rev 3.1.x

Scenarios 1 and 2 fixed

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

eTSEC 68: Rx TCP/UDP checksum checking may be incorrect while operating at low frequencies in FIFO mode

Description: In FIFO mode operation, a portion of the Rx TCP/UDP checksum checking state machine assumes that the controller has two cycles to respond to certain events on the Rx interface. If the controller runs slower than twice the Rx clock frequency, it may be unable to respond to an event and could either report a false Rx TCP/UDP checksum error (RxFCB[ETU] = 1) for a good packet, or fail to report a true TCP/UDP checksum error (RxFCB[ETU] = 0).

The false or missing checksum errors only occur on frames of size 4n + 1 byte (8-bit FIFO) or 4n + 2 bytes (16-bit FIFO), and the last byte(s) of data are to be included in the checksum calculation (note that frames truncated due to size>MAXFRM may not report a checksum error).

Impact: Rx TCP/UDP checksum checking is not supported for low clock ratios in FIFO modes (both GMII-style and encoded). Truncated frames may fail to report a checksum error.

Workaround: Option 1: Turn off Rx checksum checking by setting RCTRL[TUCSEN] = 0.

Option 2: Ensure that the minimum platform frequency to eTSEC Rx_CLK ratio with Rx checksum checking enabled in FIFO mode is greater than or equal to 4:1 (for example, 500 MHz platform frequency with 125 MHz Rx_CLK).

Option 3: Make sure that packets that are being checksummed have at least 4 bytes of data at the end of the packet that is not to be included in the checksum (such as a CRC)

Fix plan: Fixed in Rev 3.1.x

eTSEC 69: Filer does not support matching against broadcast address flag PID1[EBC]

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 70: eTSEC does not support parsing of LLC/SNAP/VLAN packets

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

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

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

Fix plan: No plans to fix

eTSEC 71: eTSEC filer reports incorrect Ether-types with certain MPLS frames

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 72: Compound filer rules do not roll back the mask

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

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

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

Table 4. Compound Rule Example for Work Arounds

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 73: Incomplete frame with error causes false CR error on next frame

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

The incomplete frames that can trigger the error are:

1. A junk frame (random data with arbitrary # of beats and no start-of-frame found)
2. A frame with preamble, start-of-frame and with no data after the start-of-frame
3. A frame with preamble-only (no start-of-frame)

Incomplete frames can occur due to collisions in half-duplex mode, or during recovery after a link down condition.

Impact: An incomplete frame with error causes a false CR (code group or CRC) error on the next frame.

Workaround: To work around this problem when the link is down, typically the PHY generates a link down interrupt. When this link down interrupt is detected, software should

1. Perform a soft_reset (MACCFG1[Soft_Reset] = 1) or a receiver reset (MACCFG1[Reset Rx Func] = 1)
2. Keep the MAC in reset until the link is up again.
3. Discard any frames received during link down.
4. Re-enable the receiver once the link is up.

Fix plan: Fixed in Rev 3.1.x

eTSEC 74: Parser does not check VER/TYPE of PPPoE packets

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

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

Workaround: None

Fix plan: Fixed in Rev 3.1.x

eTSEC 75: Back-to-back Rx frames may lose parser results of second frame

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

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

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

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

Fix plan: Fixed in Rev 2.1

eTSEC 76: RMCA, RBCA counters do not correctly count valid VLAN tagged frames

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 77: Tx errors truncate packets without error in 8-bit Encoded FIFO mode

Description: In 8-bit encoded FIFO mode, there is no error code group or error signal to indicate a Tx error. If FIFOCFG[CRCAPP] = 1, a Tx error should corrupt the appended CRC to indicate the error. The documentation of 8-bit encoded FIFO mode states that Tx FIFO under-runs cause the controller to transmit a stream of invalid bytes . Instead, the Tx simply truncates the frame without appending the CRC at all.

Impact: Transmit under-run, bus error (invalid address or data error on Tx BD or data fetch) and Tx FIFO data parity errors (see also eTSEC 61 concerning data parity errors) cause truncated frames without CRC corruption or transmission of invalid bytes.

Workaround: Enable CRC append on Tx by setting FIFOCFG[CRCAPP] = 1 and CRC checking on Rx by setting FIFOCFG[CRCCHK] = 1. The truncation of the packet at the error presents a smaller than 1 in 4 billion chance that the last four bytes of the packet match the running CRC32 calculation, ensuring that the packet is still seen as an error.

Fix plan: Fixed in Rev 3.1.x

eTSEC 78: No parser error for packets containing invalid IPv6 routing header packet

Description: If a packet with an IPv6 routing header has the “segments left” field greater than the actual number of Destination Addresses (DA) contained in the routing header, then this is an invalid packet. The erratum is that the eTSEC uses the last destination address from the routing header as part of the pseudo-header calculation for the L4 checksum Because this is really an invalid packet, the checksum should not be checked and a parse error should be flagged.

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

Workaround: Consistency checks for IPv6 routing header packets must be performed in software, or skipped.

Fix plan: Partially fixed in Rev 3.1.x

Fixed the issue of an invalid packet being flagged as having a valid checksum (RxFCB[CTU]=1).

eTSEC 79: Generation of Ethernet pause frames may cause Tx lockup and false BD close

Description: The process of generating a PAUSE control frame may cause the controller to stop transmitting (Tx lockup). Once the controller enters a Tx lockup state, only a reset of the eTSEC (toggle MACCFG1[Tx_EN]) will allow it to start transmitting frames again.

In addition to locking up, the controller may transmit two pause control frames instead of one. If this occurs, the controller erroneously sees the second pause frame as a transmitted data frame and closes the next TxBD.

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

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

Workaround:

- Option 1: Disable transmit flow control by setting MACCFG1[Tx_flow] = 0.
- Option 2: Run with a minimum platform (CCB) frequency of 500 MHz to insure correct Ethernet operation at gigabit data rates. This workaround only applies to Ver 2.1.x.
- Option 3: Use the following method to detect TX lock up and recover from it.

  Monitor Transmit Packet Counter TPKT and or Transmit Buffer Descriptor Pointers TBPTRn by software and if no change is detected in expected time, (several multiple of maximum transition time) and the Transmit halt of ring n bits in the Transmit Status Register TSTAT[THLTn] is not set, assume the transmitter is stuck by the above condition and proceed with the following recovery mechanism:

  a. Set DMACTRL[GRS] = 1 to perform a graceful receive stop
  b. Wait for IEVENT[GRSC] to be set, indicating stop complete
  c. Set DMACTRL[GTS] = 1 to perform a graceful transmit stop
  d. Wait for IEVENT[GTSC] to be set, indicating stop complete
  e. Write a 1 to IEVENT[GTSC] to acknowledge the event
  f. Clear MACCFG1[Tx_Flow]
  g. Wait 256 TX_CLK cycles
  h. Clear MACCFG1[Tx_EN]
  i. Wait 3 TX_CLK cycles
  j. Set MACCFG1[TX_EN] and MACCFG1[Tx_Flow]
  k. Set DMACTRL[GTS] = 0 to clear the graceful transmit stop
  l. Set DMACTRL[GRS] = 0 to clear the graceful receive stop

Fix plan: Fixed in Rev 3.1.x

eTSEC 80: eTSEC parser does not perform length integrity checks

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

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

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

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

Fix plan: No plans to fix

eTSEC 81: eTSEC does not verify IPv6 routing header type field

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 82: Transmission of truncated frames may cause hang or lost data

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

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

See also eTSEC 65.

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

Workaround:

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 83: L3 fragment frame files on non-existent source/destination ports

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

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

Workaround:

- Option 1: Include a filer rule to reject on PID1[IPF] at the beginning of the table.
- Option 2: Limit parsing to L2 by setting RCTRL[PRSDEP] = 01. Note that limiting parsing to L3 by setting RCTRL[PRSDEP] = 10 is not a valid work around (refer to eTSEC 66).

Fix plan: Fixed in Rev 3.1.x

eTSEC 84: Multiple BD frame may cause hang

Description: In the device reference manual, it states the following:

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

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

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

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

Fix plan: Fixed in Rev 3.1.x if multiple Tx queues enabled.

No plans to fix for single queue case.

eTSEC 85: TxBD[TC] is not reliable in 16-bit FIFO modes

Description: CRC append can be done in hardware, in software, or not at all in FIFO modes. If some frames have software CRC append, and some do not, customers may enable hardware-based CRC append on a frame-by-frame basis by setting TxBD[TC] = 1. In 16-bit FIFO modes (GMII-style or encoded), the TxBD[TC] setting for a frame is not properly pipelined, so some frames with TxBD[TC] = 1 end up transmitting without CRC appended and some frames with TxBD[TC] = 0 end up with CRC appended by the controller.

Impact: The Ethernet controller does not reliably append CRC based on TxBD[TC] in 16-bit FIFO modes.

Workaround:

- Option 1: Set FIFOCFG[CRCAPP] = 1 in 16-bit FIFO modes to force hardware append of CRC on all frames, regardless of TxBD[TC] state.
- Option 2: Set all TxBD[TC] = 0.

Fix plan: Fixed in Rev 3.1.x

eTSEC 86: eTSEC receivers may not be properly initialized

Description: When MACCFG1[Rx_EN] is enabled during system boot as part of the eTSEC port initialization sequence, the eTSEC Rx logic may not be properly initialized.

Impact: When the eTSEC Rx logic is not properly initialized, packet data may be incorrectly received . When this occurs, the first packet received (and subsequent packets) will be affected. Failure modes can include eTSEC Rx logic “deadlock”.

Workaround: During system boot, prior to setting MACCFG1[Rx_EN] in each eTSEC, perform the following:

1. Configure the eTSEC for the expected operation (for example, if use of MAC address filtering is intended, then enable address filtering; if use of the filer is intended, then enable the filer), but do not set MACCFG1[Tx_EN] and MACCFG1[Rx_EN].
2. Set MACCFG1[Loop Back].
3. Set MACCFG1[Tx_EN] and MACCFG1[Rx_EN]
4. Transmit two (2) packets and compare the received packets to the transmitted packets. If packets are not received within 10 ms, toggle MACCFG1[Rx_EN] and repeat step (4).
5. If the comparison is not equal, toggle MACCFG1[Rx_EN] and repeat step (4) to re-initialize the eTSEC Rx logic.
6. Clear MACCFG1[Loop Back] for the eTSEC.

NOTE

While in Loop Back mode, the controller must also be in Full-Duplex mode.

Fix plan: Fixed in Rev 2.1

eTSEC 87: Arbitrary Extraction cannot extract last data bytes of frame

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

The following packet and extraction types are affected:

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

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

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

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

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

Workaround: None

Fix plan: Fixed in Rev 3.1.x

eTSEC 88: False TCP/UDP and IP checksum error in FIFO mode without CRC appending

Description: Running the Ethernet controller in 8- or 16-bit FIFO mode (GMII-style or encoded) with CRC appending and checking disabled (FIFOCFG[CRCAPP] = 0 for Tx and FIFOCFG[CRCCHK] = 0 for Rx) may cause the IP or TCP/UDP checksum parsing to skip the last two bytes of the frame. This results in a false IP or TCP/UDP header checksum error because the parser may not read the data in the frame properly.

Impact: IP or TCP/UDP checksum checking, enabled through RCTRL[IPCSEN] and RCTRL[TUCSEN], may generate false errors reported in the RxFCB in FIFO modes when a CRC is not appended to the frame.

Workaround: Set FIFOCFG[CRCCHK] = 1 to enable CRC checking on Rx and enable CRC append on the corresponding Tx on the link (by setting the equivalent of FIFOCFG[CRCAPP] = 1). Having CRC in the frame ensures that the last data beat is properly aligned for IP or TCP/UDP checksum checking.

Fix plan: Fixed in Rev 3.1.x

eTSEC 89: Frames greater than 9600 bytes with TOE = 1 will hang controller

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 90: Setting RCTRL[LFC] = 0 may not immediately disable LFC

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 91: VLAN Insertion corrupts frame if user-defined Tx preamble enabled

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

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

Workaround: Use one of the following workarounds:

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

Fix plan: No plans to fix

eTSEC 92: 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.

On silicon version 2.1 only, the false parity error may cause FCS corruption on the transmitting frame, if there is one.

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 93: Transmit fails to utilize 100% of line bandwidth

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

- For revisions prior to 3.0 for single-queue operation, IPG varies between 12–15 cycles.
- For revision 3.0, erratum is fixed for single queue operation and IPG is 12 cycles.
- For multiple queue operation with fixed priority scheduling, IPG for back-to-back frames from different queues varies between 70–140 cycles.
- For multiple queue operation with round-robin scheduling, IPG for back-to-back frames from different queues is on the order of 20–40 cycles longer than multiple queue operation with fixed priority.

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

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

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

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

Fix plan: Partially fixed in Rev 3.1.x

Fixed for single queue operation and IPG is 12 cycles.

eTSEC 94: Rx packet padding limitations at low clock ratios

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

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

At low clock ratios (less than 2:1 eTSEC system clock to TSECn_TX_CLK), it is possible to overflow the receive buffer where the extra bytes are inserted before data is written to the 2 KB Rx FIFO. When this Rx buffer overflow occurs, the current Rx frame will be dropped and the subsequent frame may be passed to memory without the expected padding bytes inserted.

Impact: If the eTSEC is running at less than 2:1 eTSEC system clock to TSECn_TX_CLK ratio, the controller cannot support inserting 24 or more total bytes (from padding, and the preamble) in front of the frame data

Workaround: Limit total receive packet byte insertion via RCTRL[PAL] and Rx preamble enable to less than 24 bytes total when running at less than 2:1 eTSEC system clock:TSECn_TX_CLK ratio.

Fix plan: Fixed in Rev 3.1.x

eTSEC 95: False TCP/UDP checksum error for some values of pseudo header Source Address

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 96: Unexpected babbling receive error in FIFO modes

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

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

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

Fix plan: No plans to fix

eTSEC 97: User-defined Tx preamble incompatible with Tx Checksum

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

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

Workaround: Use one of the following workarounds:

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

Fix plan: No plans to fix

eTSEC 98: FIFO16 interface encoded mode maximum frequency is 1/4.2 platform clock

Description: The current specification for eTSEC running in FIFO16 encoded mode allows the FIFO interface to run as fast as 1/3.2 of the platform clock. For example, for a device running platform at 533 MHz, the maximum supported FIFO16 interface frequency would be 166 MHz.

At FIFO16 interface frequencies faster than 1/4.2, the Ethernet controller may fail to process incoming frames properly, leading to corrupted frame data, parser results, or controller state.

Note that GMII-style FIFO interfaces already specify a maximum interface frequency of 1/4.2 the platform clock.

Impact: The maximum interface frequency for either encoded or GMII-style mode is 1/4.2 the platform frequency, as shown in the following table:

Platform Frequency	Max FIFO16 Interface Frequency
600 MHz	142 MHz
533 MHz	126 MHz
500 MHz	119 MHz

Workaround: Run the FIFO16 interface no faster than 1/4.2 the platform clock.

Fix plan: Fixed in Rev 3.1.x

eTSEC 99: 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: Fixed in Rev 3.1.x

eTSEC 100: 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).

eTSEC 101: Magic Packet Sequence Embedded in Partial Sequence Not Recognized

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

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

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

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

- Sequence a) FF_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

  Seventh byte of 0xFF does not match next expected byte of Magic Packet Sequence (01). Pattern search restarts looking for 6 bytes of FF at byte 01.
- Sequence b)
  FF_FF_FF_FF_FF_FF_01_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...

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

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

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

- Sequence c) FF_FF_FF_FF_FF_FF_01_02_03_04_FF_FF_FF_FF_FF_FF_01...

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

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

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

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

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

Fix plan: Partially fixed in Rev 3.1.x

Sequences a) and b)—fixed

Sequence c)—no plans to fix

eTSEC 104: Rx may hang if RxFIFO overflows

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

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

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

Statistical lockup detection and recovery:

Lockup detection:

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

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

Lockup recovery:

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 105: 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. Therefore, a frame without a valid DA could erroneously cause the ethernet controller to exit Magic Packet mode. The only events that cause the MAC to go back to check for valid DA before checking for a Magic Packet sequence on new frames are:

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

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

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

Workaround: None

Fix plan: Fixed in Rev 3.1.x

eTSEC 106: Excess delays when transmitting TOE=1 large frames

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

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

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC 107: Controller may not be able to transmit pause frame during pause state

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

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

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

Workaround: None

Fix plan: Fixed in Rev 3.1.x

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

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

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

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

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

Fix plan: Fixed in Rev 3.1.x

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

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

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

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

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

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

Workaround: For GRS scenario:

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

MAX Rx reset procedure:

1) Clear MACCFG[RX_EN].

2) Wait three Rx clocks.

3) Execute workaround for the following eTSEC erratum: eTSEC receivers may not be properly initialized.

Fix plan: No plans to fix

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

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

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

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

Fix plan: Fixed in Rev 3.1.x

eTSEC-A005: Rx may hang on eTSEC3 and eTSEC4 under extreme temperature, low Vdd and heavy traffic

Description: Affected Devices: All revs and part numbers of the MPC8548(E) or MPC8547(E) with platform frequencies of 533 MHz.

Writes from either eTSEC3 or eTSEC4 to memory that require a stall due to platform bandwidth limitations can lead to a lockup of eTSEC3 and eTSEC4. Writes to memory can momentarily stall under multiple conditions that are difficult to predict. The probability of a stall increases with high eTSEC Rx utilization in combination with high platform bandwidth utilization. The problem is negatively effected by elevated device junction temperature or low operating voltage. eTSEC1 and eTSEC2 are not affected.

Impact: Heavy traffic on eTSEC3 or eTSEC4 – particularly at high temperature - may result in eTSEC3 and eTSEC4 lock up. Only a hardware reset will unlock the controller.

Workaround: Reduce platform frequency to 400MHz for full spec operation. There is no workaround for 533 MHz operation. If a lockup occurs, only a hardware reset will unlock the controller.

Fix plan: Fixed in Rev 2.1.3

PCI-Ex 35: Rx detection by transmitter does not pass high receiver impedance test

Description: The PCI Express specification high receiver impedance test (PHY Electrical Test Considerations, Revision 1.0) requires the receiver detect function to not detect a receiver when there is up to 3 nF of capacitance across a 200 kΩ resistor. The PCI Express controller only supports up to 200 pF of capacitance.

Impact: This has only been seen to cause problems with some mid-bus probes that exhibit a high capacitance.

Workaround: None

Fix plan: Fixed in Rev 3.1.x

PCI-Ex 37: Completion Timeout error disable corrupts CRS threshold error data

Description: Several attributes of enabled error conditions are written to the PEX Error Capture Status register (offset 0xE20) when an error condition is detected. If PEX_ERR_DISR[PCTD]=1 (disable detection of Completion Timeout threshold errors), then the global source ID attribute (PEX_ERR_CAP_R2[GSID]) for CRS Threshold errors will be incorrect.

Impact: An incorrect global source ID is captured in PEX Error Capture Status register for CRS Threshold errors if Completion Timeout errors are disabled.

Workaround: Enable Completion Timeout and CRS threshold error detection by keeping the default setting of PEX_ERR_DISR[PCTD]=0 and PEX_ERR_DISR[CRSTD] = 0.

Fix plan: Fixed in Rev 3.1.x

PCI-Ex 38: PCI Express LTSSM may fail to properly train with a link partner following $\overline{\text{HRESET}}$

Description: Following $\overline{\text{HRESET}}$, 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 $\overline{\text{HRESET}}$, (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: In the boot code perform the following function for both root complex and endpoint applications:

1. OR the value at CCSRBAR offset 0x0_AF00 with 0x0800_0000
2. Wait 1 ms
3. AND the value at the CCSRBAR offset 0x0_AF00 with 0xF7FF_FFFF.

This sequence resets the PCI Express controllers only. There is no workaround for this erratum when booting from PCI Express.

Fix plan: Fixed in Rev 3.1.x

PCI-Ex 39: No mechanism for recovery from hang after access to down link

Description: When its link goes down, the PCI Express controller clears all outstanding transactions with an error indicator and sends a link down exception to the interrupt controller if PEX_PME_MES_DISR[LDDD] = 0. If, however, any transactions are sent to the controller after the link down event, they will be accepted by the controller and wait for the link to come back up before starting any timeout counters (e.g. completion timeout). There is no mechanism to cancel the new transactions short of a device $\overline{\text{HRESET}}$.

Impact: New transactions sent to a PCI Express link which is down will not complete or invoke a recoverable time out until the link recovers. The outstanding transactions may cause a core or other master (e.g. DMA) hang or a core watchdog timeout.

Workaround: The problem can be mitigated by ensuring that link down exceptions are enabled (PEX_PME_MES_DISR[LDDD] = 0), and cause access to the PCI Express interface to be disabled for the duration of the link down. Note that PCI Express controller access can be via normal reads and writes (LAW/ATMU target) or by reads or writes to CONFIG_DATA in the PCI Express controller memory-mapped register space. Both types of access must be prevented.

This does not completely eliminate the problem, because there could be accesses to the PCI Express interface between the time the link goes down and the time the interrupt handler disables access to the interface. In this instance device $\overline{\text{HRESET}}$ is required.

Fix plan: Fixed in Rev 3.1.x

PCI-Ex 41: PCI Express x8 mode requires minimum platform frequency of 527 MHz

Description: Due to bandwidth requirements of the PCI Express port, a minimum platform frequency of 527 MHz is required for PCI Express to successfully operate in x8 mode. Current documentation includes the following equation for minimum platform frequency:

- For proper PCI Express operation, the CCB clock frequency must be greater than:

  500 MHz x (PCI-Express link width)/8.

When running in x8 PEX mode this equation resulted in requiring a minimum platform frequency greater than 500 MHz. This equation, no longer holds true. This equation should actually read as follows:

- For proper PCI Express operation, the CCB clock frequency must be greater than or equal to:

  527 MHz x (PCI-Express link width)/8.

When running in x8 PEX mode this equation results in requiring a minimum platform frequency greater than or equal to 527 MHz.

If PCI Express operates in x8 mode at platform frequencies below 527 MHz, many correctable errors can occur. It is possible for these correctable errors to cause the link to go down and require to be re-trained. The possibility of this occuring increases with lower platform frequencies and higher PCI Express traffic. Note that data corruption is not a symptom of this erratum.

Impact: PCI Express will not operate reliably in x8 mode when the platform frequency is between 500-527 MHz.

Workaround: Operate the platform frequency at the required frequency calculated in the above updated equation.

This equation is updated in the Hardware Specifications.

Fix plan: No plans to fix

The hardware specification documents reflect these platform frequency requirements.

PCI-Ex 42: Reads to PCI Express CCSRs or local config space temporarily return all Fs

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

The canceling of config read or write transactions should not apply to accesses to configuration, control and status registers in memory-mapped space or local PCI Express config space. Writes to memory-mapped or local PCI Express config registers are handled normally. However, due to this erratum, reads return all Fs for the duration of the link down cleanup.

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

Impact: Reads to configuration, control and status registers in the memory-mapped or config space of PCI Express return all Fs during link down cleanup or hot reset event.

During this time, writes are handled normally, though the results of the write are not verifiable.

The duration of the link down cleanup varies depending on the number and type of pending transactions. Cleanup for pending outbound transactions takes just a few cycles, regardless of the source. Pending inbound transactions wait for all responses from targets (data response for reads, or successful arbitration to the target queue for writes) before completing cleanup.

Once all pending transactions have been cleared, new CCSR accesses and local config return to normal operation.

Note that because of PCI-Ex 39 , any new accesses to PCI Express, including config reads to off-chip registers, will wait until the link resumes normal operation to complete. The config access state machine is serialized, so a config read to an off-chip register will prevent access to local CCSRs or local config operations while the link remains down.

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

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

Fix plan: Fixed in Rev 3.1.x

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Fix plan: Fixed in Rev 3.1.x

SEC 5: AES-CTR mode data size error

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

Impact: None

Workaround: Use one of the following options:

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

Fix plan: Fixed in Rev 2.1

SEC 6: Single descriptor SRTP error

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

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

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

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

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

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

By running these descriptors back to back with no interrupt in between, the data operated upon by the second descriptor will likely be fetched from L2, thereby reducing the memory bus utilization associated with the second descriptor.

Fix plan: Fixed in Rev 2.1

SEC 7: AES-CCM ICV checking write-back error

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

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

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

Fix plan: No plans to fix

SEC 8: Kasumi hardware ICV checking does not work

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

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

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

Fix plan: No plans to fix

SEC-A001: Channel Hang with Zero Length Data

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

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

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

Fix plan: No plans to fix

SRIO 39: Serial RapidIO atomic operation erratum

Description: Applications using **lwarx/stwcx** instructions in the core to compete for a software lock or semaphore with a device on RapidIO using read atomic set, clr, inc, or dec in a similar manner may falsely result in both masters seeing the lock as "available." This could result in data corruption as both masters try to modify the same piece of data protected by the lock.

It is possible that a read atomic set, clr, inc, or dec from a RapidIO master may be bypassed by a read transaction on behalf of a lwarx instruction from the core. This can result in the core seeing the lock "not set", while allowing the read atomic operation to complete and report success back to the RapidIO master. It also never clears the reservation made by the core just prior to issuing the read transaction on behalf of its lwarx instruction. The reason for this lack of ordering between the two read transactions is that the transactions may queue in either of two separate read command queues of the memory scheduling block. Ordering is not enforced between each of the read queues. Note that ordering in both read queues is enforced with respect to the write queue.

Impact: Both the core and Rapid IO master may attempt to modify a data structure at the same time that is protected by a software lock/semaphore. This can only occur if the core is using lwarx/stwcx instructions and the RapidIO master is using read atomic set, clr, inc, or dec transactions to check and obtain ownership of the lock.

Workaround: For any application where the core is issuing **lwarx/stwcx** instructions to the same cacheline address that a RapidIO master is sending read atomic set, clr, inc, or dec transactions, one of two software solutions below should be followed. The first solution retains maximum performance of the memory subsystem by temporarily disabling one of the read command queues during the lock sequence. The second solution doesn't require the core to precede its **lwarx/stwcx** routine with any special code sequence but requires permanently disabling one of the read command queues.

First Solution—code sequence to be run:

1. Read CCSR offset 0x01010 (EEBPCR register of the ECM) into GPR A
2. Bit-wise OR GPR A with 0x0004_0004
3. Write the result back to CCSR offset 0x01010
4. Read CCSR offset 0xE0F14 (DDRTQDCR0 register of Global Utilities) into GPR A
5. Bit-wise AND GPR A with 0xF1FF_ E3FF
6. Bit-wise OR GPR A with 0x0000_0001
7. Write the result back to CCSR offset 0xE0F14
8. Read CCSRBAR
9. SYNC
10. Run a loop for 5 microseconds
11. Read CCSR offset 0xE0F14 into GPR A
12. Bit-wise OR GPR A with 0x0600_0000
13. Bit-wise AND GPR A with 0xFFFF_FFFE
14. Write the result back to CCSR offset 0xE0F14
15. Enter the **lwarx/stwcx** routine...

If the core ever attempts to take ownership of this lock again, the same sequence must be repeated.

Second Solution:

A second software solution is to simply set bits 13 and 29 of CCSR offset 0x01010 (EEBPCR register of the ECM) during initialization and leave them set indefinitely. This may slightly degrade overall system performance.

Fix plan: No plans to fix

SRIO 41: Serial RapidIO Packets with errors are not ignored by the controller while in input-retry-stopped state

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

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

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

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

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

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

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

Fix plan: No plans to fix

SRIO 42: Message unit cannot generate messages with priority 0

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

OMnDATR[DTFLOWLVL]

00	SRIO transaction priority of 1
01	SRIO transaction priority of 1
10	SRIO transaction priority of 2
11	Reserved

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

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

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

Fix plan: No plans to fix

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

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

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

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

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

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

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

Fix plan: Fixed in Rev 3.1.x

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

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

Output port transmit enable:

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

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

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

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

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

Examples of non-compliant scenarios:

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

Impact: When PmCCSR[OPE] = 0

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

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

- Outbound maintenance request packets will not be transmitted.

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

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

Fix plan: No plans to fix

DDR 13: Memory contents may not be retained during $\overline{\text{HRESET}}$ sequence

Description: It may be desirable for customers to have the DDR controller enter self refresh mode and $\overline{\text{HRESET}}$ the part shortly after, while retaining contents of memory. However, it is possible that CKE will not be driven active low during $\overline{\text{HRESET}}$, bringing the DRAM out of self refresh.

There are pin-sampled signals that may erroneously place the DDR into a test mode if they are not set to a valid state when $\overline{\text{HRESET}}$ is asserted. The DDR controller should drive the MCKE[0:3] pins active low throughout and after $\overline{\text{HRESET}}$. However, when this test mode is entered, the DDR drivers will be inactive, and the MCKE[0:3] pins will be released to high impedance. This test mode will be entered if a value of 0xf is presented on LA[28:31] during $\overline{\text{HRESET}}$, or if a value of 0x0 is presented on MSRCID[2] during $\overline{\text{HRESET}}$. The MCKE[0:3] pins will remain released to high impedance until LA[28:31] and MSRCID[2] are set correctly for pin sampling.

Impact: The DRAMs may erroneously exit self refresh mode if MCKE is released to high impedance and transitions above the minimum AC switching voltage level.

Workaround: Depending upon the board application, it may be possible to use active components during $\overline{\text{HRESET}}$ to ensure that MCKE[0:3] will remain low throughout $\overline{\text{HRESET}}$.

Fix plan: No plans to fix

DDR 14: The automatic CAS-to-Preamble feature of the DDR controller can calibrate to incorrect values

Description: When the DDR controller executes the automatic CAS-to-preamble logic, it will be possible that the controller will calibrate to an erroneous value. In addition, there will not be an error reported in the ERR_DETECT register, and there is no valid way to tell if the calibrated value is valid.

Impact: The automatic CAS-to-preamble function of the DDR controller can fail, which could lead to data corruption on the DDR interface if this feature is enabled.

Workaround: The automatic CAS-to-preamble feature should not be enabled. Instead the CAS-to-preamble should be calculated (using examples shown in application note AN2583 from Freescale).

Fix plan: No plans to fix

DDR 15: Automatic calibration hardware may calibrate to an invalid driver impedance

Description: The DDR controller will typically calibrate to half-strength drive mode (highest impedance setting) when calibrating the DDR IO drive strength. This same issue has been observed when calibrating the drive strength via software. Since this calibration uses an 18 Ω resistor on the board, it is not expected to resolve to the highest impedance setting.

Impact: This calibration was only intended for use with full-strength drivers. There was not a calibration option for half-strength drive mode. Therefore, applications using half-strength drive mode are not affected.

Workaround: The automatic driver calibration should not be used: DDRCDR[DHC_EN] should be cleared.

- For Full-strength Mode, the default driver impedance setting used by the controller will force the nominal 18 Ω setting.
- Half-strength mode can also be enabled via setting the DDR_SDRAM_CFG[HSE] bit in the DDR controller's memory mapped space, forcing a nominal 36 Ω setting.

Customers should not need to write the impedance overrides in DDRCDR register.

Fix plan: Fixed in Rev 2.1

DDR 16: On-die termination at the DDR IOs has been measured 75 Ω too high

Description: The DDR IOs provide termination options of 75 Ω and 150 Ω. Silicon measurements show about 150 Ω and 225 Ω, respectively.

Impact: The termination at the DDR IOs will be inaccurate. By setting the 75 Ω option, one can still get termination of about 150 Ω. However, there is no way to get 75 Ω.

Workaround: When trying to obtain 150 Ω termination, the 75 Ω termination option can be set by clearing DDRCDR[ODT]. However, there is no workaround to obtain 75 Ω termination. Note that this issue has been present on all previous revisions of the device. Customers who designed using simulation results should not have to alter impedance settings via software in fixed silicon, as the lines will simply be better balanced using fixed silicon.

Fix plan: Fixed in Rev 2.1

DDR 17: DDR performance monitoring and tracing functionality does not work

Description: Performance Monitor and Watchpoint/Trace functionality does not work correctly with respect to certain types of DDR-related information.

Impact: The following functionality is not supported in 2.0 silicon only:

DDR Performance Monitor events:

Counter 1: Event 0

Counter 2: Event 0, 1

Counter 3: Event 0, 1, 2, 60

Counter 4: Event 0, 1, 3, 4

Counter 5: Event 0, 2, 56

Counter 6: Event 0, 1, 2, 5

Counter 7: Event 1, 4, 57

Counter 8: Event 0, 2

Reference Events 11, 12, 13, 14, 1

Watchpoint/Trace buffer:

when “internal DDR SDRAM interface” is selected (WMCR1[IFSEL] or TBCR1[IFSEL] = b’001).

Workaround: Do not use the aforementioned DDR performance monitor events. The same Watchpoint/ Trace buffer information can be obtained from the e500 Coherency Module Dispatch selection (WMCR1[IFSEL] or TBCR1[IFSEL] = 3b’000).

Fix plan: Fixed in Rev 2.1

DDR 18: Automatic data initialization and DLL resets to DRAM are not performed correctly if CS2 and CS3 are interleaved

Description: CS2 and CS3 can be interleaved together by setting DDR_SDRAM_CFG[BA_INTLV_CTL] to 7'bx1xx0x0. In this mode, the DDR controller may operate incorrectly for 2 different features.

First, the DDR controller will not initialize DRAM data properly if DDR_SDRAM_CFG_2[D_INIT] is set. If D_INIT is set with CS2 and CS3 interleaved together, then the memory spaced defined by CS2 and CS3 will not be initialized.

Second, the DDR controller may not issue DLL reset commands to CS2 and CS3 when exiting self refresh.

Note that this feature is typically enabled when DDR_SDRAM_CFG_2[DLL_RST_DIS] is cleared. If CS0 and CS1 are both disabled via CSn_CONFIG[CS_n_EN] (and CS2 and CS3 are interleaved together), then only CS3 will receive the DLL reset command when self refresh is exited.

Note that neither of these issues will be present if all chip selects are enabled and CS0-CS3 are interleaved together.

Impact: There are 2 results that can be observed from this erratum. If the first scenario listed above is present, then the memory space defined by CS2 and CS3 will not be initialized properly when DDR_SDRAM_CFG_2[D_INIT] is set. If the second scenario listed above is present, then the DLL reset command will not be issued as expected to CS2. It is not expected that this will cause any issues with DDR2 memories. DDR2 JEDEC specifications state that the DRAM's DLL is automatically disabled when entering self refresh, and the DLL is automatically reenabled when exiting self refresh. Although it would be possible for some vendors to vary, the DLL reset should not be required by the DRAMs when exiting self refresh. The DLL reset feature was originally added to support DDR1 memories. It appears that DDR1 memories will typically only require the DLL reset when the frequency is changed, but this scenario should still be avoided if possible to prevent any potential issues.

Workaround: There are several workarounds for this erratum. The preferred workaround will be to disable interleaving between CS2 and CS3. CS0 and CS1 can still be interleaved together. In addition, CS0 and CS3 could still be interleaved together without any issues.

If it is still preferred to interleave CS2 and CS3 together, then one of two workarounds can be used for initializing memory. First, software could be used to initialize memory (i.e., via the DMA) instead of using DDR_SDRAM_CFG_2[D_INIT]. In addition, the memory controller could be enabled without CS2 and CS3 interleaved while DDR_SDRAM_CFG_2[D_INIT] is set. After D_INIT is cleared by the hardware, CS2 and CS3 could then be programmed to be interleaved together (via DDR_SDRAM_CFG[BA_INTLV_CTL]). The memory space defined by the CS2_BNDS register would then need to be updated to include the entire memory space for CS2 and CS3. During this entire sequence, software would need to guarantee that no other transactions are issued to memory.

Other than disabling interleaving between CS2 and CS3, the only other way to workaround the DLL reset issue is to ensure that either CS0 or CS1 is also enabled (via CSn_CONFIG[CS_n_EN]).

Fix plan: No plans to fix

DDR 19: DDR IOs default receiver biasing may not work across voltage and temperature

Description: The DDR IO receiver biasing is controlled through settings in an engineering use only register. The current default settings may not work at cold temperature. The worst case condition for this erratum is TJ = 0 degC, GVDD = GVDD(min), VDD = VDD(max). When a failure occurs, a DDR input latches an incorrect value.

Impact: The DDR interface may fail if the default receiver biasing value is not overridden.

Workaround: Write register at CCSRBAR offset 0xE_0F24 with a value of 0x9000_0000 for DDR2 and a value of 0xA800_0000 for DDR1 before enabling the DDR controller. This will set the receiver to an acceptable bias point.

Fix plan: Fixed in Rev 2.1

Note: the workaround for the Rev 2.0 device must be removed when running the Rev 2.1 device.

DDR 20: MCKE signal may not function correctly at assertion of $\overline{\text{HRESET}}$

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

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

The POR configuration pins used to enter test mode are LA[28:31] and MSRCID[2]. If the LA[28:31] are driving a value of 0b1111 at the time of $\overline{\text{HRESET}}$ assertion, then MCKE[0:3] may be released to high impedance. If MSRCID[2] is driving a zero at the time of $\overline{\text{HRESET}}$ assertion, then MCKE[0:1] may be released to high impedance or drive a 1.

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

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

Option 1

Note that the following DDR DEBUG registers are used in this option:
- D2—offset is CCSRBAR + DDR_OFFSET + 0xF04
- D3—offset is CCSRBAR + DDR_OFFSET + 0xF08

At assertion of $\overline{\text{HRESET}}$, perform an alternative DDR controller initialization sequence. This clears the DRAM state machines and allows them to operate properly. Before this sequence is implemented, do not enable any DDR LAWBAR entries. Details of alternative sequence are as follows:
1. Configure DDR registers as is done in normal DDR configuration. Do not set DDR_SDRAM_CFG[MEM_EN].
2. Set reserved bit EEBACR[3] at offset 0x1000.
3. Before DDR_SDRAM_CFG[MEM_EN] is set, write DDR_SDRAM_CFG_2[D_INIT].
4. Before DDR_SDRAM_CFG[MEM_EN] is set, write D3[21] to disable data training.
5. Wait 200 μs (as described in the “DDR SDRAM Initialization Sequence” section in the device reference manual).
6. Set DDR_SDRAM_CFG[MEM_EN].
7. Poll DDR_SDRAM_CFG_2[D_INIT] until it is cleared by hardware.
8. Clear D3[21] to re-enable training.
9. Set D2[21] to force the data training to run.
10. Poll on D2[21] until it is cleared by hardware.

    After this step, there are two options that can be followed if ECC is enabled before continuing on to step 10 . If DDR ECC is not utilized, continue to step 10 . Sub-Option 1 requires a calculated delay. Sub-Option 2 does not require the delay, but it is not supported for applications with DDR interleaving enabled.

    Sub-Option 1:

    a. Wait calculated delay

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

Delay = 400 ms/Gbytes × max memory size. For 32-bit data buses, multiply this number by 2.

Example: Assume a 64-bit DDR2 with total memory size = 1 Gbyte.

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

b. Set DDR_SDRAM_CFG_2[D_INIT].
c. Poll on DDR_SDRAM_CFG_2[D_INIT] until it is cleared by hardware, then the system can proceed.
d. Enable any DDR LAWBAR entries and proceed to step 10 .

Sub-Option 2:

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

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

Option 2

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

Option 3

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

Fix plan: No plans to fix

GEN 10: Device may fail DDR pins during IEEE Std 1149.1™ EXTEST mode

Description: During IEEE Std 1149.1™ EXTEST mode, the DDR I/O drivers may be disabled due to un-initialized logic. These controls are correctly configured during functional POR, which is not a requirement for EXTEST testing.

Impact: Using the EXTEST mode the DDR drivers may be randomly disabled. This may effect customer's manufacturing board tests, causing false failures on the DDR interconnect tests, while the device DDR pins are driving.

Workaround: Freescale has provided an updated BSDL definition file [Revision R2B]. This updated BSDL is modified to define HRESET as a compliance pin for all JTAG testing. This removes the HRESET pin from the interconnect testing, but will allow all the DDR signals to be correctly tested.

Fix plan: No plans to fix

GEN 11: Some pins do not meet 500V CDM ESD criteria

Description: CDM (Charged Device Model) ESD testing has shown that some SerDes pins do not meet the 500V CDM ESD criteria. The following pins are impacted:

- SD_TX[7:0], SD_TX_B[7:0], SD_RX[7:0], SD_RX_B[7:0], SD_REF_CLK, SD_REF_CLK_B, SD_PLL_TPD, SVDD, XVDD, AVDD_SRDS, SD_IMP_CAL_TX, SD_IMP_CAL_RX, SD_PLL_TPA
- GND pins U27, L21, L23, N22, P20, R23, T21, U22, V20, W23, Y21, K28, L24, L26, N24, N27, P25, R28, T24, T26, U24, V25, W28, Y24, Y26, A24, AB25, AC28

Impact: None

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

Fix plan: Fixed in Rev 3.1.x

GEN 13: CPU write to platform failures in all core, platform, and SYSCLK frequency combinations

Description: AVDD filter resistors must be increased on new builds to avoid data corruption for CPU writes to the platform.

Impact: A problem originally thought to only impact systems that are running with a core frequency less than 1200 MHz, or a platform frequency less than 333 MHz, or a SYSCLK frequency greater than 133 MHz has been observed on systems operating at higher core frequencies, higher platform frequencies and lower SYSCLK frequencies. The issue manifests itself as corrupted data on CPU writes to the platform. The issue is exacerbated by high temperature and high VDD voltage.

Workaround: Systems are restricted to the following configurations:

1. Core is limited to minimum frequency of 800 MHz.
2. Platform (CCB) is limited to minimum frequency of 333 MHz
3. SYSCLK is limited to maximum frequency of 133 MHz

Affected systems must change the value of the PLL power supply filtering resistors (Section 21.2.1 of the hardware specification). The filter resistors must be increased from 10 ohm to 180 ohm for the AVDD_CORE filter resistor, and 150 ohm for the AVDD_PLAT filter resistor. The tolerance of these resistors must be +/-5% or better.

Additionally, systems with an operating core frequency less than 1200 MHz must limit SYSCLK frequency to 100 MHz maximum.

Fix plan: Fixed in Rev 3.1.x

GEN 14: CPU-only reset may result in a failure in the platform logic

Description: The device provides the ability to assert the $\overline{\textit{core\_reset}}$ signal to the processor. Once done, the possibility exists that the platform logic may fail.

Impact: If $\overline{\textit{core\_reset}}$ is asserted by setting PIR[P0] or DBCR0[RST], the possibility exists that the platform logic may fail.

Workaround: The core must be reset via a full hard reset of the device.

Fix plan: No plans to fix

SERDES 1: SerDes lane power down function may cause training to fail.

Description: SerDes Controller register 1 (SRDSCR1) provides a mechanism to power down individual lanes of the SerDes port. The HW specification recommends that unused lane(s) should be powered down by setting the appropriate bit(s) in this register. It was discovered that powering down lanes [via this mechanism] may cause the link not to train.

Impact: Link training may fail causing the SerDes port to be inoperative.

Workaround: Do not set any SDRSCR1 control register bits.

Fix plan: No plans to fix

Documentation will be updated to remove this feature from the MPC8548 Reference Manual. The MPC8548 Hardware Specification will be updated to remove the recommendation.

LBIU 3: LBIU Transactions when clock divider LCRR[CLKDIV] is changing

Description: Functionally the LBIU should not accept transactions when the LBIU is changing the clock divider LCRR[CLKDIV]. Any read request from the core or the platform to the LBIU block should stall during the CLKDIV changing period. This is supposed to be invisible to software. The stall is supposed to clear once LBIU resynchronizes its clock counters, and the read (instruction fetches) from LBIU should resume. However, due to the erratum, the internal logic does not start to block the LBIU requests immediately, potentially letting a transaction start that will fail due to the LBIU resynchronization process.

Impact: Any local bus transaction may fail during LBIU resynchronization process when the clock divider [CLKDIV] is changing.

Workaround: Ensure there is no transaction on the local bus for at least 100 microseconds after changing clock divider LCRR[CLKDIV]. User can run instructions from L2SRAM or DDR memory when changing LBIU clock divider LCRR[CLKDIV]. This erratum applies to both LBIU PLL bypass mode and PLL enable mode.

Fix plan: No plans to fix

LBIU 4: $\overline{\text{LGTA}}$/LUPWAIT assertion in PLL-bypass mode misrepresented

Description: In the Local Bus section of the Hardware Specification document, the timing diagram figures for PLL-bypass mode show that the $\overline{\text{LGTA}}$/LUPWAIT signal is latched on the falling edge of the internal launch/capture clock signal. This is a misrepresentation of the device functionality as $\overline{\text{LGTA}}$/LUPWAIT is actually latched on the subsequent rising edge of the internal launch/capture clock signal.

Impact: Asserting $\overline{\text{LGTA}}$/LUPWAIT for only the falling edge of the internal launch/capture clock signal in PLL-bypass mode will result in the local bus controller not registering the $\overline{\text{LGTA}}$/LUPWAIT input.

Workaround: **Option 1:** Asserting the $\overline{\text{LGTA}}$/LUPWAIT signal for an additional LCLK cycle in PLL-bypass mode will guarantee the local bus controller registering the $\overline{\text{LGTA}}$/LUPWAIT input correctly.

OR

Option 2: Assert the $\overline{\text{LGTA}}$/LUPWAIT signal for only the rising edge of the internal launch/capture clock signal in PLL-bypass mode.

Fix plan: No plans to fix

The Hardware Specification will be updated to show that the $\overline{\text{LGTA}}$/LUPWAIT signal is latched 1/2 LCLK cycle later on the rising edge of the internal launch/capture clock signal.

LBIU 5: UPM does not have indication of completion of a RUN PATTERN special operation

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

The UPM is expected to have an indication of when a special operation is completed. The UPM will see MxMR[MAD] increment when a write to or read from UPM array special operation completes. However, the UPM does not have any status indication of completion of the Run Pattern special operation.

The following scenario could be affected by this erratum:

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

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

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

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

Workaround: None

Fix plan: No plans to fix

PIC 4: PIC soft reset does not clear MSIRn registers correctly

Description: The MSIRn registers are not cleared by writing to the GCR[RST].

Impact: It is possible to falsely detect an interrupt after writing to the GCR[RST] bit to clear the PIC.

Workaround: To perform a reset command, software should set the GCR[RST] bit and immediately follow this with reads to each of the MSIRn registers. Reads to the MSIRn registers will automatically clear the register contents. Data from each of the MSIRn reads should be discarded.

Fix plan: No plans to fix

PIC 5: MSIMR De-featured

Description: MSIIR register can set bits in MSIR when disabled in MSMIR. Interrupt will not be forwarded to its destination when masked using the MSIMR register. When a message shared interrupt is asserted while masked using the MSIMR register the respective interrupt will not be forwarded when subsequently unmasked.

Impact: Interrupt will not be forwarded to its destination.

Workaround: In order to mask the message shared interrupt without loss use the MSK bit in the MSIVPRn. Note that the MSIMR register defaults out of reset to zeros which is enabled. Software should prevent writes to this register.

Fix plan: Documentation update

This register will be de-featured from reference manual.

PIC 6: MER, Interrupt will not be forwarded to destination

Description: Interrupt will not be forwarded to its destination when masked using the MER register. When a message interrupt is asserted while masked using the MER register the respective interrupt will not be forwarded when subsequently unmasked.

Impact: Interrupt will not be forwarded to its destination in Ver.1.x and Ver.2.0.

Workaround: No workaround. In order to mask the interrupt without loss, use the MSK bit in the MIVPRn. Note that the MER register defaults out of reset to zeros which is disabled. Therefore, during the initialization, set certain MER bit(s) to enable desired message interrupts.

Fix plan: No plans to fix

PIC 7: PCI Express MSI other than interrupt 0 not supported via hardware

Description: PCI Express MSI memory write is defined as a 32-bit write to the address location in the Message Address Register of the MSI Capability Register. The (little-endian) data format of the write is 31:16 - all zeroes, 15:0 from the Message Data Register of the MSI Capability Register, with lower bits modified as necessary to indicate the particular message.

The MPIC implements MSI via the Message Shared Interrupt Index Register (MSIIR), which is big-endian with two fields: Shared Interrupt Register Select (SRS - bits 24:26) and Interrupt Bit Select (IBS - bits 27:31).

The PCI Express Root Complex logic swaps the bytes of inbound writes to big-endian memory space, so the msi_data[31:24] is written to MSIIR[24:31], msi_data[23:16] to MSIIR[16:23], msi_data[15:8] to MSIIR[8:15], and msi_data[7:0] to MSIIR[0:7]. Since msi_data[31:16] are defined as all zeroes, MSIIR[SRS] and MSIIR[IBS] are always set to zero on an MSI write, and all standard MSIs trigger interrupt 0 (MSIR0[SH0]=1, MSISR[S0]=1). The contents of msi_data[15:0] are lost.

Impact: The only MSI that can be triggered through the standard PCI Express hardware mechanism and configuration is interrupt 0.

Workaround: **Option 1:** If no other devices or mechanisms write to the 1 MB memory-mapped register region defined by CCSRBAR through PCI Express, use an inbound ATMU window with the attributes set as follows:

1. Set PEXIWBAR0 to some memory region unused by PCI Express
2. Enable an inbound ATMU (example using window 1):
   a. PEXIWBAR1 = previous value of PEXIWBAR0
   b. PEXIWTAR1[8:31] = CCSRBAR[8:23]||8'b0
   c. PEXIWAR1[0:31] = 0xC0F4_4013 (Enable, No prefetch, Local Memory No snoop, 1 MB window)

Please notice that workaround option 1 might not be compatible with future versions of the IP.

Option 2: Use a software or other programmable mechanism to generate the 32-bit MSI memory write with MSI[31:24] set to the value of MSI[7:0]. Duplicating the MSI data in the LSB and MSB allows the workaround to be compatible with future versions of the IP which correct the definition and implementation of MSIIR.

Fix plan: No plans to fix

PM 1: Some local bus events are not counted correctly in the performance monitor

Description: The followings events are not counted correctly:

- Bank 1–8 hits (chip-select)
- Request granted to ECM port
- Cycles atomic reservation for ECM port is enabled

Impact: These local bus events can not be used in the performance monitor.

Workaround: None

Fix plan: No plans to fix

PM 2: A performance monitor time base transition event will not freeze the performance monitor counters and may not cause an exception

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

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

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

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

Fix plan: No plans to fix

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

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

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

Workaround: Use one of the following workarounds:

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

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

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

Fix plan: Fixed in Rev 3.1.x

I2C 3: I2C boot sequencer cannot issue snoopable writes

Description: The I2C boot sequencer cannot issue snoopable writes. Boot sequencer transactions are therefore not presented on the CCB.

Impact: There is no possible way to use the I2C boot sequencer to initialize regions of the L2 configured as SRAM. Note that only being able to issue Non-snoopable transactions causes no cache coherency issues since the core is not permitted to perform its initial boot vector fetch until after the boot sequencer completes its initialization process.

Workaround: None

Fix plan: Fixed on Rev 2.1

DMA 1: $\overline{\text{DMA\_DACK}}$ bus timing violation when operating in external DMA master mode

Description: The specification requires the external DMA master signal $\overline{\text{DMA\_DACK}}$ to be held for at least three system clocks. The DMA violates this requirement. The $\overline{\text{DMA\_DACK}}$ signal is directly associated with the channel MR[CS] bit while in external mode. The MR[CS] is “simulated” while in external mode by hardware setting the bit when DMA_DREQ is asserted, and clearing the bit when the channel wins arbitration. In normal DMA mode, a transfer in progress is determined by seeing bit MR[CS] and bit MR[CB] asserted together. In external mode this is not the case, the MR[CS] bit is asserted to short of a time and does not truly reflect transfer in progress. This problem causes the $\overline{\text{DMA\_DACK}}$ to assert for a very short time. In addition to the problem above, there is no logic to force the assertion of $\overline{\text{DMA\_DACK}}$ to be at least three system clock cycles.

Impact: $\overline{\text{DMA\_DACK}}$ held for six CCB_clocks.

Workaround: None.

Fix plan: Partially fixed in Rev 3.1.x

If the CCB Clock PLL Ratio is not configured to either the 9:1 or 20:1 setting, this erratum is fixed.

DMA 2: Transfer error reported for wrong channel

Description: The DMA controller has resources that are shared between all channels. Each channel is given a time period in the shared resources corresponding to the value of the bandwidth control specified in MR[BWC]. The last write transaction corresponding to the transfer of a block (specified by the byte count register in the channel) is a write that requires a response from the target port (WRFTP). This type of write is referred to here as an WRFTP. While a channel that sent its last write data is waiting for the write response, another channel is allowed to start using the shared resources.

When the WRFTP gets an error response, it is the channel that is active in the shared resources that will get the transfer error bit set, not the channel that is waiting for the response of the WRFTP transaction. Sources of error responses for an WRFTP are:

- The write gets a translation error from an outbound ATMU translation window at the target port (Serial RapidIO, PCI Express).
- The WRFTP translates to a non-posted write on PCI Express, and the non-posted write receives an error response from the attached device (WRFTP translation is controlled by ATMU configuration).
- The WRFTP translates to an NWRITE_R on Serial RapidIO and the write receives an error response from the attached device (WRFTP translation is controlled by ATMU configuration).

Note that an error response on a DMA read will set the transfer error bit in the correct channel. This problem is limited to getting an error on WRFTP response.

Impact: The wrong channel could have Transfer Error set. The actual failing channel will complete normally, when data may not actually have been written to the destination successfully. When the Transfer error bit is set for one channel, software will have to assume that it could be have been caused by any other channel.

Workaround: A few work arounds have been identified with varying performance and software impact:

- Do not configure the ATMUs as described above, or
- When a channel completes a block transfer or descriptor chain, check that no other channel has its transfer error set, or
- Use one channel at a time. Not very practical since it reduces the DMA to a one channel DMA.

Fix plan: Fixed in Rev 3.1.x

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

Affects: DUART

Description: Previously DUART 1

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

PCI 4: PCI-X inbound reads resulting in a split completion response can cause a corrupted response

Description: PCI-X inbound reads that result in a split completion response from the PCI-X logic during or after PCI-X outbound traffic has occurred can cause a corrupted response from the PCI-X device. If the requester of the inbound read transaction retries the split completion response, the PCI-X logic may corrupt the outbound command queue. Instead of sending the last split completion response, the PCI-X logic may send out a Dual Address Cycle (DAC) with an upper 32-bit address of 0xFFFF_FFFF. The lower 32-bit address will contain the attribute phase information of the original read request. If the requester does not retry the split completion response, this erratum will not apply. Clock speed will not affect this issue. Also, outbound traffic must be present sometime during this situation in order to see this errata. Please keep in mind that outbound traffic that has already completed may still affect our internal logic and corrupt the last split completion response.

Impact: Per the PCI-X specification, only a PCI-X bridge is allowed to retry a split completion response. This issue will only affect these systems.

Workaround: To work around this issue, the requestor of the original read transaction must accept all split completion responses. Another possible workaround is to request smaller blocks of data so the logic will not respond with a split completion response.

Fix plan: No plans to fix

PCI 5: Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang

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

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

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

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

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

PCI-X operation is not affected by this erratum.

Workaround: A PCI target device could avoid asserting STOP# during the last beat of a PCI memory write transaction that is greater than one data beat and crosses a cacheline boundary. It could assert STOP# during the last data beat of the cacheline or not assert STOP# at all. A software workaround for this problem is to set the PCI Latency Timer Register (offset 0x0D) to zero. A value of zero is the reset value for this register, so if this register is kept unmodified after reset, it will prevent the PCI IP block from ever combining writes.

A debug bit, bit 10 of the PCI Bus Function Register (address 0x44) has been identified as another workaround. This bit, MDS (Master disable streaming), when set, will disable the combining of crossing cacheline boundary requests into one burst transaction,. Therefore, it can prevent the errata scenario from occurring. When this bit 10 is set, customer can program the PCI Latency Timer to achieve full cacheline burst even with one request in the PCI pipe.

Fix plan: No plans to fix

PCI 6: PCI/PCI-X erroneous error detection

Description: PCI/X ERR_DR[OWMSV] and ERR_DR[ORMSV] (offsets 0x0_8E00, 0x0_9E00) bits can erroneously set and may trigger an interrupt if capturing and reporting of these events are enabled.

Impact: ERR_DR[OWMSV] cannot be used to alert the user of an outbound write memory space violation. ERR_DR[ORMSV] cannot be used to alert the user of an outbound read memory space violation. This bug results in less coverage for programming errors, but does not affect the functionality of the controller when the checking is disabled.

Workaround: Set ERR_CAP_DR[27] and ERR_CAP_DR[28] to disable OWMSV, ORMSV error capture. Also clear ERR_EN[27] and ERR_EN[28] to disable OWMSV, ORMSV error reporting. It is also permissible to immediately return from interrupt when ERR_DR[OWMSV] or ERR_DR[ORMSV] is set.

Fix plan: Fixed in Rev 2.1

PCI 7: Asynchronous mode PCI1 and PCI2 input hold violation

Description: PCI1 and PCI2 fail input hold by 250 ps in asynchronous mode

Impact: A board which has less than 250 ps of hold time when driving data to the MPC8548E and running PCI1 or PCI2 in asynchronous mode may fail.

Workaround: Provide 250 ps or greater input hold to the MPC8548E when running PCI1 or PCI2 in asynchronous mode.

Fix plan: Fixed in Rev 2.1

PCI 8: Device reports wrong PCI Vendor ID

Description: The device provides the wrong PCI Vendor ID. The value returned is 0x1057, instead of the correct value of 0x1957.

Impact: The device indicates Motorola as the PCI Vendor instead of Freescale.

Workaround: None.

Fix plan: No plans to fix

PCI 9: Master-Abort issued incorrectly for outbound DAC with subtractive decode

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

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

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

Fix plan: No plans to fix

JTAG 2: TMS requires hold time beyond the fall of TCK

Description: If the SAMPLE/PRELOAD or EXTEST instruction is the current instruction in the JTAG TAP, and the JTAG state machine is moving into the UPDATE-DR state, TMS must be held past the falling edge of TCK.

Impact: This requirement violates the IEEE 1149.1 spec which only requires TMS to be valid at the rise of TCK.

The boundary scan update register will not update, and interconnect testing will fail.

This primarily affects 3rd party JTAG and hardware tool vendors because this only affects the SAMPLE/PRELOAD and EXTEST JTAG instructions, as these are the only ones required to update the boundary register.

Workaround: **Option 1:** Hold the TMS signal 2ns past the falling edge of TCK.

Option 2: If the SAMPLE/PRELOAD or EXTEST instruction is the current instruction in the JTAG TAP, and the JTAG state machine is in the UPDATE-DR state, move to the SELECT-DR state instead of to RUN-TEST-IDLE.

Moving from UPDATE-DR to RUN-TEST-IDLE requires TMS to change from a ‘1’ to a ‘0’. In this case, care must be taken to ensure that TMS is held at ‘1’ for 2 ns past the falling edge of TCK before changing to the ‘0’ value.

If the JTAG tool has the option to move to the SELECT-DR state instead of the RUN-TEST-IDLE state from the UPDATEDR state, then the value on the TMS pin does not change, because TMS will be a ‘1’ moving into the UPDATE-DR state and a ‘1’ moving into the SELECT-DR state. Keeping TMS at a ‘1’ value will satisfy the hold time requirement. Then, go through the IR route to go back to the RUN-TEST-IDLE state.

Fix plan: Fixed in Rev 2.1

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: MPC8548ECE
Rev. 6
06/2012