

MPC8572E Chip Errata

This document details all known silicon errata for the MPC8572E PowerQUICC™ III devices.

The following table provides a revision history for this document.

Table 1. Revision History

Revision	Date	Substantive Changes
K	02/2010	Added eLBC-A001, eTSEC-A002, IEEE1588-A001, and DMA-A001. Added sentence to eTSEC 56 workaround.
J	11/2009	Added CPU-A001, eTSEC 59, eTSEC-A001. Updated DDR9. Updated disposition for eTSEC 8 Updated eTSEC 13 workaround -corrected PID1 to PID0.
I	07/2009	Updated DDR 2.
H	06/2009	Added row for Rev 2.1 in Table 2. Added column for Rev 2.1 in Table 3. Updated eTSEC 21. Updated disposition for eTSEC 10, eTSEC 57.
G	05/2009	Updated workaround for eTSEC 46. Added DDR 14, eTSEC 55, IEEE 1588_10 (previously referred to as eTSEC56), FEC 3, SEC 7, SRIO 3, eTSEC 56, eTSEC 57, eTSEC 58. Added Rev 2.0 information to Table 2, "Revision Level to Part Marking Cross-Reference." Updated GEN 1. Updated disposition for PCI-Ex 3. Updated work around and disposition for DDR 8. Updated work around for UART 1.

Revision	Date	Substantive Changes
F	12/2008	<p>Added row for Rev 2.0 in Table 2.</p> <p>Added column for Rev 2.0 in Table 3.</p> <p>Added CPU 5, DDR 9, DDR 10, eTSEC 53, eTSEC 54, GEN 1, PCI-Ex 5, SRIO 2, UART 1.</p> <p>Updated Work Around for eTSEC 10.</p> <p>Updated eTSEC 50.</p> <p>Corrected disposition for eTSEC 17.</p> <p>Updated disposition for DDR 2, DDR 8, GPIO 1, eTSEC 2, eTSEC 3, eTSEC 4, eTSEC 6, eTSEC 8, eTSEC 9, eTSEC 11, eTSEC 12, eTSEC 13, eTSEC 14, eTSEC 15, eTSEC 16, eTSEC 18, eTSEC 20, eTSEC 23, eTSEC 24, eTSEC 25, eTSEC 26, eTSEC 28, eTSEC 30, eTSEC 31, eTSEC 32, eTSEC 33, eTSEC 34, eTSEC 35, eTSEC 36, eTSEC 37, eTSEC 38, eTSEC 41, eTSEC 42, eTSEC 44, eTSEC 45, eTSEC 46, eTSEC 51, eTSEC 52, IEEE 1588_1, IEEE 1588_3, IEEE 1588_5, IEEE 1588_6, IEEE 1588_7, IEEE 1588_9</p>
E	8/2008	<p>Updated Table 2.</p> <p>Updated disposition for DDR 2, eTSEC 8, eTSEC 17.</p> <p>Added DDR 8, eLBC 2, eLBC 3, eTSEC 47 through eTSEC 52, I²C 2, SEC 5, SEC 6.</p> <p>Updated SEC 5</p>
D	4/2008	<p>Updated Table 2, Revision Level to Part Marking Cross-Reference.</p> <p>Added CPU 2 through CPU 4.</p> <p>Added DDR 5 through DDR 7.</p> <p>Added PCI-Ex 4.</p> <p>Added eTSEC 42 through eTSEC 46.</p> <p>Added IEEE 1588_7 through IEEE 1588_9.</p> <p>Added GPIO 1.</p> <p>Updated eTSEC 19, eTSEC 22, eTSEC 38, eTSEC 40, IEEE 1588_1.</p> <p>Updated disposition for eTSEC 5, eTSEC 6, eTSEC 7, eTSEC 19, eTSEC 22, eTSEC 29, I2C1, IEEE_1588_2, JTAG 1, PCI-Ex 1, PCI-Ex 2, PME 1.</p>
C	11/2007	<p>Deleted eTSEC 33 because it was a duplicate of eTSEC 28.</p> <p>Assigned the number eTSEC 33 to a new erratum.</p> <p>Added CPU 1, DDR 3, DDR 4, DMA 1, eLBC 1, eTSEC 33 through eTSEC 41, IEEE 1588_1, IEEE 1588_4-6, IEEE 1588_UM 1, I2C 1, PCI-Ex 1 through PCI-Ex 3, PME 1, FEC 2 and SRIO 1.</p> <p>Added and updated IEEE 1588_2-3, JTAG 1.</p> <p>Updated DDR 2, eTSEC 3, eTSEC 6, eTSEC 8, eTSEC 10, and eTSEC 29.</p> <p>Updated disposition for DDR 1, eTSEC 5, eTSEC6, eTSEC9 through eTSEC11, eTSEC13, eTSEC19, eTSEC32, and SEC 1 through SEC 4.</p>
B	6/2007	<p>Added DDR 2, eTSEC 12 through eTSEC 33, and FEC 1.</p> <p>Updated eTSEC 11.</p>
A	3/2007	Initial NDA release

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

Table 2.

MPC8572E Revision	e500 V2 Core Revision	Processor Version Register	Device Marking	System Version Register
1.0	3.0	0x8021_0030	00M44J	With Security Engine 0x80E8_0010
			60M44J	Without Security Engine 0x80E0_0010
1.1	3.0	0x8021_0030	01M44J	With Security Engine 0x80E8_0011
			61M44J	Without Security Engine 0x80E0_0011
1.1.1	3.0	0x8021_0030	02M44J	With Security Engine 0x80E8_0011
			62M44J	Without Security Engine 0x80E0_0011
2.0	3.0	0x8021_0030	00M20X	With Security Engine 0x80E8_0020
			60M20X	Without Security Engine 0x80E0_0020
2.1	3.0	0x8021_0030	01M20X	With Security Engine 0x80E8_0021
			61M20X	Without Security Engine 0x80E0_0021

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

Note that Rev 2.0 is offered only in samples and will not be offered as a qualified production device.

Table 3. Summary of Silicon Errata and Applicable Revision

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	1.1.1	2.0	2.1
CPU						
CPU 1	"mbar MO = 1" instruction fails to order caching-inhibited guarded loads and stores	No plans to fix	Yes	Yes	Yes	Yes
CPU 2	Single-precision floating-point zero value may have the wrong sign	No plans to fix	Yes	Yes	Yes	Yes
CPU 3	Branch Fails to Decrement CTR in Presence of D-cache Parity Error	No plans to fix	Yes	Yes	Yes	Yes
CPU 4	Incorrect Value May be Loaded Into SRR0 on ITLB Miss When Executing icbtlis	No plans to fix	Yes	Yes	Yes	Yes
CPU 5	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	Yes
CPU-A001	Flash-Invalidation of TLB1 with "mtspr MMUC-SR0" may fail	No plans to fix	Yes	Yes	Yes	Yes
DDR						
DDR 1	DDR controller may enter an illegal state when operating in 32-bit bus mode with 4-beat bursts	No plans to fix	Yes	Yes	Yes	Yes
DDR 2	DDR controller may not function properly with DDR3 registered DIMMs.	Partially fixed in Rev 1.1.1	Yes	Yes	Yes	Yes
DDR 3	Signal integrity issues may be observed on the ECC pins of the DDR interface if a 32-bit bus is used	Fix attempted in Rev 1.1.1	Yes	No	No	No
DDR 4	The automatic CAS-to-Preamble feature of the DDR controller can calibrate to incorrect values	No plans to fix	Yes	Yes	Yes	Yes
DDR 5	Invalid data on MECC[5] when debug information is sourced onto the MECC[0:5] pins	No plans to fix	Yes	Yes	Yes	Yes
DDR 6	A false address parity error may be detected when DDR_SDRAM_CFG[MEM_EN] is set	No plans to fix	Yes	Yes	Yes	Yes
DDR 7	The memory controller does not use the correct default impedances for the IOs in DDR3 mode	No plans to fix	Yes	Yes	Yes	Yes
DDR 8	MCKE signal may not function correctly at assertion of HRESET	No plans to fix	Yes	Yes	Yes	Yes
DDR 9	DDR controller may not work with dual-ranked DDR3 registered DIMM if write leveling is enabled	No plans to fix	Yes	Yes	Yes	Yes
DDR 10	DDR to CCB clock ratios of greater than 1.5:1 can result in DDR configuration register corruption	Fixed in Rev 2.0	Yes	Yes	No	No
DDR 11	Asserting panic interrupt via IRQ_OUT the DDR controller may violate the JEDEC specifications	No plans to fix	Yes	Yes	Yes	Yes
DDR 12	The DDR controller may fail the write leveling sequence for DDR3	No plans to fix	Yes	Yes	Yes	Yes
DDR 13	Change in DDRCLK input frequency range	No plans to fix	Yes	Yes	Yes	Yes

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	1.1.1	2.0	2.1
DDR 14	Data may become corrupted for DDR3 32-bit bus mode	No plans to fix	Yes	Yes	Yes	Yes
DMA						
DMA 1	DMA_DACK bus timing violation when operating in external DMA master mode	No plans to fix	Yes	Yes	Yes	Yes
DMA-A001	Uncorrectable data read errors for DMA transfers may also cause DMA lockup or additional detectable data corruption	No plans to fix	Yes	Yes	Yes	Yes
eLBC						
eLBC 1	UPM does not have indication of completion of a Run Pattern special operation	No plans to fix	Yes	Yes	Yes	Yes
eLBC 2	LTEATR and LTEAR may show incorrect values under certain scenarios	No plans to fix	Yes	Yes	Yes	Yes
eLBC 3	Some local bus events are not counted correctly in the performance monitor	No plans to fix	Yes	Yes	Yes	Yes
eLBC-A001	Simultaneous FCM and GPCM or UPM operation may erroneously trigger bus monitor timeout	No plans to fix	Yes	Yes	Yes	Yes
eTSEC						
eTSEC 1	Frame is dropped with collision and HALF-DUP[Excess Defer] = 0	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 2	WWR bit Anomaly	Fixed in Rev. 2.0	Yes	Yes	No	No
eTSEC 3	Parsing of tunneled IP packets not supported	Fixed in Rev. 2.0	Yes	Yes	No	No
eTSEC 4	RQUEUE[EXn] bits have no effect	Fixed in Rev. 2.0	Yes	Yes	No	No
eTSEC 5	Some combinations of Tx packets may trigger a false Data Parity Error (DPE)	Fixed in Rev. 1.1.1	Yes	No	No	No
eTSEC 6	eTSEC Data Parity Error (DPE) does not abort transmit frames	See errata text	Yes	Yes	No	No
eTSEC 7	eTSEC half duplex receiver packet corruption	Fixed in Rev. 1.1.1	Yes	No	No	No
eTSEC 8	May drop Rx packets in non-FIFO modes with lossless flow control enabled	Fixed in Rev. 2.0	Yes	Yes	No	No
eTSEC 9	Parser continues parsing L4 fields when RCTRL[PRSDEP] set for L2 and L3 fields only	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 10	Fetches with errors not flagged, may cause livelock or false halt	Partially fixed in Rev 2.0	Yes	Yes	Yes	Yes
eTSEC 11	Rx TCP/UDP checksum checking may be incorrect while operating at low frequencies in FIFO mode	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 12	Parsing of MPLS label stack or non-IPv4/IPv6 label not supported	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 13	Filer does not support matching against broadcast address flag PID1[EBC]	Fixed in Rev 2.0	Yes	Yes	No	No

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	1.1.1	2.0	2.1
eTSEC 14	eTSEC does not support parsing of LLC/SNAP/VLAN packets	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 15	eTSEC filer reports incorrect Ether-types with certain MPLS frames	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 16	Compound filer rules do not roll back the mask	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 17	Incomplete frame with error causes false CR error on next frame	Fixed in Rev 1.1.1	Yes	No	No	No
eTSEC 18	Parser does not check VER/TYPE of PPPoE packets	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 19	RMCA, RBCA counters do not correctly count valid VLAN tagged frames	Fixed in Rev 1.1.1	Yes	No	No	No
eTSEC 20	Tx errors truncate packets without error in 8-bit Encoded FIFO mode	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 21	No parser error for packets containing invalid IPv6 routing header packet	Partially fixed in Rev 2.0	Yes	Yes	Yes	Yes
eTSEC 22	Generation of Ethernet pause frames may cause Tx lockup and false BD close	Fixed in Rev 1.1.1	Yes	No	No	No
eTSEC 23	eTSEC parser does not perform length integrity checks	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 24	eTSEC does not verify IPv6 routing header type field	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 25	Transmission of truncated frames may cause hang or lost data	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 26	L3 fragment frame files on non-existent source/destination ports	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 27	Multiple BD frame may cause hang	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 28	TxBD[TC] is not reliable in 16-bit FIFO modes	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 29	Arbitrary Extraction Un-extractable Bytes	Fixed in Rev 1.1.1	Yes	No	No	No
eTSEC 30	Back-to-back IPv6 routing headers not supported by parser	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 31	RxBD[TR] not asserted during truncation when last 4 bytes match CRC	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 32	VLAN insertion and extraction cannot be used in FIFO mode	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 33	Arbitrary Extraction cannot extract last data bytes of frame	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 34	L2 arb extract shifted with shim headers not multiple of 4 bytes	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 35	False TCP/UDP and IP checksum error in FIFO mode without CRC appending	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 36	Frames greater than 9600 bytes with TOE = 1 will hang controller	Fixed in Rev 2.0	Yes	Yes	No	No

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	1.1.1	2.0	2.1
eTSEC 37	Setting RCTRL[LFC] = 0 may not immediately disable LFC	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 38	False parity error at Tx startup	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 39	VLAN Insertion corrupts frame if user-defined Tx preamble enabled	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 40	Transmit fails to utilize 100% of line bandwidth	Partially fixed in Rev 2.0	Yes	Yes	Yes	Yes
eTSEC 41	eTSEC4 ECNTRL[GMIIM] bit does not set in RGMII mode	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 42	False TCP/UDP checksum error for some values of pseudo header Source Address	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 43	User-defined Tx preamble incompatible with Tx Checksum	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 44	Magic packet does not wake device from a SLEEP state	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 45	Rx packet padding limitations at low clock ratios	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 46	Controller stops transmitting pause control frames	Fixed in Rev 2.0	No	Yes	No	No
eTSEC 47	ECNTRL[AUTOZ] not guaranteed if reading MIB counters with software	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 48	Unexpected babbling receive error in FIFO modes	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 49	Half-duplex collision on FCS of Short Frame may cause Tx lockup	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 50	Magic Packet Sequence Embedded in Partial Sequence Not Recognized	Partially fixed in Rev 2.0	Yes	Yes	Yes	Yes
eTSEC 51	MAC: Malformed Magic Packet Triggers Magic Packet Exit	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 52	Receive pause frame with PTV=0 does not resume transmission	Fixed in Rev. 2.0	Yes	Yes	No	No
eTSEC 53	Rx may hang if RxFIFO overflows	Fixed in Rev 2.0	Yes	Yes	No	No
eTSEC 54	TxBD polling loop latency is 1024 bit-times instead of 512	No plans to fix	No	No	Yes	Yes
eTSEC 55	Misfiled Packets Due to Incorrect Rx Filter Set Mask Rollback	No plans to fix	No	No	Yes	Yes
eTSEC 56	Excess delays when transmitting TOE=1 large frames	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 57	Data corruption may occur in SGMII mode	Plan to fix in Rev 2.1	Yes	Yes	Yes	No
eTSEC 58	Incorrect frame data in L3+L4-only or L4-only parsing for FIFO mode	No plans to fix	Yes	Yes	Yes	Yes
eTSEC 59	Controller may not be able to transmit pause frame during pause state	No plans to fix	Yes	Yes	Yes	Yes

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	1.1.1	2.0	2.1
eTSEC-A001	MAC: Pause time may be shorter than specified if transmit in progress	No plans to fix	Yes	Yes	Yes	Yes
eTSEC-A002	GRS may fail to complete after receiving a 1- or 2-byte frame	No plans to fix	Yes	Yes	Yes	Yes
E1588						
IEEE1588_1	1588 reference clock limited to 1/2 controller core clock in asynchronous mode	Fixed in Rev 2.0	Yes	Yes	No	No
IEEE1588_2	IEEE 1588 not supported in SGMII mode	Fixed in Rev 1.1.1	Yes	No	No	No
IEEE1588_3	1588 reference clock pulse required between writes to TMR_ALARMn_L and TMR_ALARMn_H	Fixed in Rev 2.0	Yes	Yes	No	No
IEEE1588_4	1588 alarm fires when programmed to less than current time	Partially fixed in Rev 2.0	Yes	Yes	Yes	Yes
IEEE1588_5	IEEE 1588 not supported in Dual-Clock TBI mode	Fixed in Rev 2.0	Yes	Yes	No	No
IEEE1588_6	Use of asynchronous 1588 reference clock may cause errors	Fixed in Rev 2.0	Yes	Yes	No	No
IEEE1588_7	Cannot use inverted 1588 reference clock when selecting eTSEC system clock as source	Fixed in Rev 2.0	Yes	Yes	No	No
IEEE1588_8	Odd prescale values not supported	No plans to fix	Yes	Yes	Yes	Yes
IEEE1588_9	FIPER's periodic pulse phase not realigned when the 1588 current time is adjusted	Fixed in Rev 2.0	Yes	Yes	No	No
IEEE1588_10	IEEE 1588 accuracy can be adversely impacted in systems using multiple unsynchronized gigabit Ethernet ports	No plans to fix	Yes	Yes	Yes	Yes
IEEE1588-UM1	TMR_ALARMn_L register must be written before TMR_ALARMn_H	Reference manual update	Yes	Yes	Yes	Yes
IEEE1588-A001	Missing or incorrect received frame timestamp values when 1588 time-stamping is enabled	No plans to fix	Yes	Yes	Yes	Yes
FEC						
FEC 1	MII, half duplex collision causes Tx frames to lose data	No plans to fix	Yes	Yes	Yes	Yes
FEC 2	FEC: Rx_FIFO overflow due to system busy may cause receiver to hang	No plans to fix	Yes	Yes	Yes	Yes
FEC 3	FEC Management Data Clock is faster than IEEE maximum	Fixed in Rev 2.0	Yes	Yes	No	No
GEN						
GEN 1	External master cannot reset core	See errata text	Yes	Yes	No	No
GPIO						
GPIO 1	Invalid data read from GPDAT bits corresponding to pins configured as outputs	Fixed in Rev 2.0	Yes	Yes	No	No
I2C						

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	1.1.1	2.0	2.1
I2C 1	I2C boot sequencer cannot issue snoopable writes	Fixed in Rev 1.1.1	Yes	No	No	No
I2C 2	Enabling I ² C could cause I ² C bus freeze when other I ² C devices communicate	No plans to fix	Yes	Yes	Yes	Yes
JTAG						
JTAG 1	Boundary scan test on SerDes transmitter pins needs special requirement in compliant to IEEE 1149.1 specification	Fixed in Rev 1.1.1	Yes	No	No	No
PEX						
PCI-Ex 1	Completion Timeout error disable corrupts CRS threshold error data	Fixed in Rev 1.1.1	Yes	No	No	No
PCI-Ex 2	PCI Express LTSSM may fail to properly train with a link partner following HRESET	Fixed in Rev 1.1.1	Yes	No	No	No
PCI-Ex 3	No mechanism for recovery from hang after access to down link	Enhanced in Rev 1.1.1	Yes	Yes	Yes	Yes
PCI-Ex 4	Reads to PCI Express CCSRs or local config space temporarily return all Fs	No plans to fix	Yes	Yes	Yes	Yes
PCI-Ex 5	PCI Express x8 mode is not supported at platform frequencies of 400-527 MHz	The hardware specification documents reflect these platform frequency requirements.	Yes	Yes	Yes	Yes
PME						
PME 1	DXCM Parity Malfunction may cease operation of PME	Fixed in Rev 1.1.1	Yes	No	No	No
SEC						
SEC 1	Security registers from offset 0x3_1510 to 0x3_1518 cannot be accessed with 4-byte accesses	No plans to fix	Yes	Yes	Yes	Yes
SEC 2	Protocol descriptor hang conditions	No plans to fix	Yes	Yes	Yes	Yes
SEC 3	AES-GCM IV Length Restriction	No plans to fix	Yes	Yes	Yes	Yes
SEC 4	TLS_SSL_Block_Inbound HMAC Error	No plans to fix	Yes	Yes	Yes	Yes
SEC 5	Non-compliant implementation of deterministic pseudo-random number generator	No plans to fix	Yes	Yes	Yes	Yes
SEC 6	Limitations on Custom Modes of CRC	No plans to fix	Yes	Yes	Yes	Yes
SEC 7	Kasumi hardware ICV checking does not work	No plans to fix	Yes	Yes	Yes	Yes
SRIO						
SRIO 1	Accessing SRIO memory mapped space causes the device to hang when SRIO is disabled via POR configuration	No plans to fix	Yes	Yes	Yes	Yes
SRIO 2	Serial RapidIO Packets with errors are not ignored by the controller while in input-retry-stopped state	No plans to fix	Yes	Yes	Yes	Yes

Errata	Name	Projected Solution	Silicon Rev.			
			1.0	1.1.1	2.0	2.1
SRIO 3	Message Unit cannot generate messages with priority 0	No plans to fix	Yes	Yes	Yes	Yes
DUART						
UART 1	BREAK detection triggered multiple times for a single break assertion	No plans to fix	Yes	Yes	Yes	Yes

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

Description:	<p>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.</p> <p>Assume the following instruction sequence:</p> <ul style="list-style-type: none">• stw caching-inhibited, guarded address A• mbar MO = 1• lwz caching-inhibited, guarded address B <p>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.</p>
Impact:	<p>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.</p> <p>This failure is limited to caching-inhibited loads bypassing the "mbar MO = 1" barrier. The "mbar MO = 1" instruction correctly enforces the ordering of caching-inhibited stores, and the ordering of cacheable stores.</p>
Workaround:	<p>Use "mbar MO = 0" to ensure that caching-inhibited guarded loads do not bypass the memory barrier.</p>
Fix plan:	<p>No plans to fix</p>

CPU 2: Single-precision floating-point zero value may have the wrong sign

Description:	When performing single-precision floating-point operations that produce a result of zero, the sign of the zero value may be incorrect.
Impact:	Single-precision floating-point operations that result in zero may not be compatible with IEEE Std 754™.
Workaround:	None
Fix plan:	No plans to fix

CPU 3: 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 4: Incorrect Value May be Loaded Into SRR0 on ITLB Miss When Executing icbtlis

Description:	When executing the icbtlis , 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 icbtlis instructions if CT != 0.
Workaround:	<p>This problem can be avoided by executing isync after icbtlis. The program must ensure that no ITLB Miss can occur between the execution of the icbtlis and the execution of the isync. This implies that the isync should reside on the same page as the icbtlis.</p> <p>A sequence of icbtlis 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 icbtlis in this sequence and the isync at the end of the sequence.</p> <p>There is no need to avoid asynchronous interrupts between the execution of icbtlis and the following isync. The occurrence of the interrupt has the same effect as the isync and prevents the problem.</p> <p>In a multicore SoC, steps should be taken to ensure that the page containing the icbtlis and the isync instructions cannot be invalidated during execution of the icbtlis/isync sequence. In particular, one must ensure that the page cannot be invalidated during this sequence by some other processor executing a tlbivax.</p>
Fix plan:	No plans to fix

CPU 5: 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:	<p>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.</p> <p>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.</p>
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: <ul style="list-style-type: none">• 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

DDR 1: DDR controller may enter an illegal state when operating in 32-bit bus mode with 4-beat bursts

Description:	When operating with a 32-bit bus, it is recommended that DDR_SDRAM_CFG[8_BE] is cleared when DDR2 memories are used. This forces the DDR controller to use 4-beat bursts when communicating to the DRAMs. However, an issue exists that could lead to data corruption when the DDR controller is in 32-bit bus mode while using 4-beat bursts.
Impact:	If the DDR controller is operating in 32-bit bus mode with 4-beat bursts, then the controller may enter into a bad state. All subsequent reads from memory is corrupted. Four-beat bursts with a 32-bit bus only is used with DDR2 memories. Therefore, this erratum does not affect DDR3 mode.
Workaround:	To work around this issue, software must set DEBUG_1[31] in DDR memory mapped space (CCSRBAR offset + 0x2f00 for DDR_1 and CCSRBAR offset + 0x6f00 for DDR_2).
Fix plan:	No plans to fix

DDR 2: DDR controller may not function properly with DDR3 registered DIMMs.

Description: DDR3 registered DIMMs have configurable registers located on the DIMM. These registers can be written to by asserting both chip selects simultaneously while using some of the MA and MBA pins to select the register, field, and value to write. Although these registers are not completely defined yet, some of the features that will be supported will be output inversion enable, output floating enable, drive strength settings for the output of the register, delay shift for certain signals at the output of the register, 1T/3T timing, etc.

There are three issues with the RCWs:

1. Rev 1.0 only: It is possible that the DDR controller writes unexpected data to these "control words" before the DDR controller is enabled.
2. Self-refresh mode is not supported with registered DIMMs with multiple ranks. Setting `DDR_SDRAM_CFG_2[FRC_SR]` will not put the DRAM into self-refresh mode, and may corrupt the RCW.
3. The DDR controller does not support writes to these control words (RCWs). It appears that the DIMMs' default values work for some cases, but the inability to set the drive-strength may be particularly problematic.

Impact: The DDR controller may corrupt the RCWs on the registered DIMMs before the memory controller is enabled (Rev 1.0 only) or when attempting to enter self refresh on multi-rank registered DIMMs. This may cause the DIMMs to function in an undesired way, depending upon the final definition of these registers.

Self refresh mode is not supported for multi-rank registered DIMMs.

Furthermore, the controller does not have a method to allow it to write known data to these registers. This could preclude the use of some DIMMs, depending upon the final definitions and requirements.

Workaround: Self refresh should not be used with multi-ranked, registered DDR3 DIMMs. In addition, user should confirm that the default RCW values of any registered DDR3 DIMMs are acceptable.

Work around for Rev.1.0 only:

Rev 1.0 is the only revision of silicon susceptible to RCW corruption during the initialization sequence. However, there is a software work around that would allow the controller to prevent clocks from switching until the chip selects are actively driven. This will prevent corruption of the RCWs during initialization sequence.

Proposed work around:

- The controller starts toggling clocks (MCK/MCK) once any chip select has been enabled. Until then, the clocks simply are driven active high.
- The chip selects are in non-impedance mode until `DDR_SDRAM_CFG[MEM_EN]` has been written.
- After configuring all timing registers, `DDR_SDRAM_MD_CNTL` should be used to force MCKE low. Without enabling any chip selects, `DDR_SDRAM_CFG[MEM_EN]` can be set while `DDR_SDRAM_CFG[BI]` and `DDR_SDRAM_CFG[MEM_HALT]` are set to allow the chip selects to be driven active. `DDR_ZQ_CNTL[ZQ_EN]` and `DDR_WRLVL_CNTL[WRLVL_EN]` should remain cleared.

- Now, the chip selects are driven active high while MCKE is still low and MCK/MCK is idle.
- Next, the chip selects should be enabled via CSn_CONFIG[CSn_EN]. This allows MCK/MCK to begin toggling. No writes to the control words of the register should have been performed, because MCS was driven active high before the clocks were running.
- Finally, the DRAM initialization sequence to configure all MRS registers may be performed by using the DDR_SDRAM_MD_CNTL register to assert MCKE and write all MRS registers.
- If ZQ calibration is used, it may now be enabled via DDR_ZQ_CNTL[ZQ_EN]. Then, ZQ calibration may be forced by writing a "1" to DEBUG_1[1] (CCSRBAR offset + 0x2f00 for DDR_1 and CCSRBAR offset + 0x6f00 for DDR_2). This forces a ZQCL command to be issued to each chip select. Finally, DDR_SDRAM_CFG[MEM_HALT] should be cleared by software.
- Unfortunately, write leveling cannot be used with this work around. However, write leveling may not be required for DDR3-800 with registered DIMMs, depending upon the routing on the DIMMs.
- It is expected that there will be much less MCK/MCK routing variations to each discrete device on a registered DIMM.

Note that the default RCW values must still be used in this method.

Fix plan:

Partially fixed in Rev 1.1.1.

Issue #1 (unexpected values written to RCW before the DDR controller is enabled) is fixed in Rev 1.1.1. No plans to fix for Issues #2 and #3.

DDR 3: Signal integrity issues may be observed on the ECC pins of the DDR interface if a 32-bit bus is used

Description:	The termination for the ECC pins is not controlled correctly when the DDR controller is configured in 32-bit bus mode. Some of the ECC pins do not receive proper termination during reads from memory. This could lead to signal integrity issues on the DDR interface.
Impact:	The DDR interface could experience data corruption on the ECC byte during reads if a 32-bit bus is used.
Workaround:	There is no valid work around for this bug when 32-bit bus is used. ODT could be enabled at the memory to prevent overshoot/undershoot during reads, but the termination should optimally be enabled at the memory controller during reads to prevent signal integrity issues.
Fix plan:	Fix attempted in Rev 1.1.1

DDR 4: The automatic CAS-to-Preamble feature of the DDR controller can calibrate to incorrect values

Description: The DDR controllers' automatic CAS to Preamble calibration feature currently not revealed in the device reference manual Rev.0 may fail.

When this feature is enabled by setting the register bit field of TIMING_CFG_2[CPO] to 5'b11111, the DDR controller should calibrate the TIMING_CFG_2[CPO] setting that software would otherwise need to set. However, it is possible that the DDR controller might fail the calibration sequence due to this erratum.

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: Note the following register offsets:

DEBUG_1 register address = CCSRBAR offset 0x2F00 (for DDR Controller 1) or 0x6F00 (for DDR Controller 2);

DEBUG_2 register address = CCSRBAR offset 0x2F04 (for DDR Controller 1) or 0x6F04 (for DDR Controller 2);

DEBUG_3 register address = CCSRBAR offset 0x2F08 (for DDR Controller 1) or 0x6F08 (for DDR Controller 2);

1. Write all DDR memory mapped registers, except DDR_SDRAM_CFG[MEM_EN]. Then, the following sequence should be run
2. Clear DDR_SDRAM_CFG_2[ODT_CFG].
3. Write DEBUG_1 -> 0x0001_0000
4. Write DEBUG_3 -> 0x0000_0400
5. Set DDR_SDRAM_CFG[MEM_EN]
6. Poll for DEBUG_2[30]. After this bit is set by hardware, proceed to the next step.
7. Write DEBUG_1 -> 0x0000_0000
8. Set DDR_SDRAM_CFG_2[ODT_CFG] to the appropriate value (probably to assert internal ODT during reads).
9. Write DEBUG_3 -> 0x0000_0000
10. Write DEBUG_2 -> 0x0000_0400
11. Poll for DEBUG_2[30]. After this bit is set by hardware, proceed to the next step.
12. The controller should now be ready for normal operation.

Fix plan: No plans to fix

DDR 5: Invalid data on MECC[5] when debug information is sourced onto the MECC[0:5] pins

Description:	A debug mode can be used to multiplex certain debug information (source ID and data valid) onto the MECC pins. When this mode is used, the debug information intended for MECC[5] (data valid) is not correctly asserted during reads to memory. Note that the MSRCID and MDVAL pins still show correct information.
Impact:	There is no impact in functional mode, but this could affect debug capabilities for some systems.
Workaround:	The MSRCID and MDVAL pins can be used to view this debug information instead of the MECC pins.
Fix plan:	No plans to fix

DDR 6: A false address parity error may be detected when DDR_SDRAM_CFG[MEM_EN] is set

Description:	<p>Address parity is enabled by setting DDR_SDRAM_CFG_2[AP_EN] before the memory controller has been enabled. When DDR_SDRAM_CFG[MEM_EN] is set, the controller begins generating address parity while monitoring the error signal sent back from the registered DIMMs.</p> <p>However, the controller may detect a false address parity error when the memory controller is first enabled, because the register could be in a state where it is driving the parity error signal active low.</p>
Impact:	<p>ERR_DETECT[APE] are erroneously set by hardware, and an interrupt is propagated if ERR_INT_EN[APEE] is set.</p>
Workaround:	<p>Before setting DDR_SDRAM_CFG[MEM_EN], set ERR_DISABLE[APED]. After setting DDR_SDRAM_CFG[MEM_EN], issue a read back to DDR_SDRAM_CFG. Then, clear ERR_DISABLE[APED] if error reporting is desired for address parity.</p>
Fix plan:	<p>No plans to fix</p>

DDR 7: The memory controller does not use the correct default impedances for the IOs in DDR3 mode

Description: If the memory controller is operating in full-strength mode without software overrides or hardware calibration via DDRCDR_1/DDRCDR_2, then it uses a 40-Ω target for the IO drivers. If the memory controller is operating in half-strength mode (DDR_SDRAM_CFG[HSE] = 1), then the default values used for the IOs have a much higher impedance. The full strength default should have targeted 19–20 Ω, and the half-strength default should have targeted about 40 Ω. In addition, the DDR controller does not allow driver calibration to train with an 18–20 Ω precision resistor. Instead, it only supports training in DDR3 mode with higher impedances (~ 40 Ω).

Impact: If incorrect driver impedances are used, then setup/hold violations may be present on the DDR interface. The inability to calibrate to an 18-Ω value is not considered a large impact, as it is expected that customers train to the impedance desired for the data (about 40 Ω).

Workaround: If training will not be used, then the following values should be programmed into the software overrides for each group in the DDRCDR_1/DDRCDR_2 registers:

- If half-strength (~ 40 Ω) is desired for a particular group, set the DSO_*PZ and DSO_*NZ fields to 4'b0111 (also set DSO_*_EN).
- If full-strength (~ 19–20 Ω) is desired for a particular group, set the DSO_*PZ and DSO_*NZ fields to 4'b1010 (also set DSO_*_EN).

Note that leaving DDR_SDRAM_CFG[HSE] cleared without any overrides sets all groups to the 40-Ω targeted setting. However, this may be targeted to a much lower impedance setting (~20 Ohms) in future revisions. If training is used for the half-strength values, then the full-strength groups can still have software overrides as described above.

Fix plan: No plans to fix

DDR 8: 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 pin used to enter test mode is MSRCID[2]. If MSRCID[2] is driving a zero at the time of $\overline{\text{HRESET}}$ assertion, then MCKE[0:1] on DDR Controller #1 may drive a one.

The DDR controller should drive the MCKE[0:1] pins active low throughout and after $\overline{\text{HRESET}}$ assertion. However, when the DDR controller enters a test mode, the DDR MCKE driver is released to high impedance or driven high, preventing the MCKE[0:1] pins from being driven low immediately after $\overline{\text{HRESET}}$ assertion.

Note that when the controller is configured for DDR3 mode, the DRAMs have a $\overline{\text{RESET}}$ pin that can be controlled by the $\overline{\text{HRESET}}$ pin on our device to reset the DRAMs everytime an $\overline{\text{HRESET}}$ is asserted on our device. This eliminates the JEDEC requirement for MCKE to be driven low at $\overline{\text{HRESET}}$ assertion.

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

Do not use chip select 0 or 1 on DDR controller #1.

Option 2

At assertion of $\overline{\text{HRESET}}$ perform an alternative DDR controller initialization sequence for each utilized controller. 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:

Note the following DEBUG registers:

- D1 offset is CCSRBAR + DDR_OFFSET + 0xF00
- D2 offset is CCSRBAR + DDR_OFFSET + 0xF04
- D3 offset is CCSRBAR + DDR_OFFSET + 0xF08
- D4 offset is CCSRBAR + DDR_OFFSET + 0xF14
- D5 offset is CCSRBAR + DDR_OFFSET + 0xF18

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 μ s (as described in section "DDR SDRAM Initialization Sequence," of the applicable 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. If combining this workaround with DDR4 workaround, perform steps in section below entitled "Additional steps for DDR4/DDR8 combined workaround." Otherwise, continue.
10. Set D2[21] to force the data training to run.
11. 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 12 . If DDR ECC is not utilized, enable the DDR LAWBARs and continue to step 12 . 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/Gbyte \times max DDR1 or DDR2 controller memory size

For 32-bit data buses, multiply this number by 2.

Example: assume 64-bit DDR2, memory size = 1 Gbyte on DDR1 controller, 1 Gbyte on DDR2 controller

Delay = 400 ms/Gbyte \times 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 12.

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.

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

Additional steps for DDR4/DDR8 combined workaround

1. Clear DDR_SDRAM_CFG_2[ODT_CFG]

2. Set D1[15]
3. Set TIMING_CFG_2[CPO] to 5'b11111 to use auto CAS to preamble
4. Set D4 to a value of 0x9f9f9f
5. Set D5 to a value of 0x9f9f9f
6. Set D2[20] to force the automatic CPO to train
7. Poll on D2[20] until it is cleared by hardware
8. Clear D1[15]
9. Set DDR_SDRAM_CFG_2[ODT_CFG] to the appropriate value (probably to assert internal ODT during reads)
10. Resume original workaround.

Option 3

Use an active component (for example, CPLD) to drive MCKE signals to the DRAMs. For example, a CPLD or FPGA can be used to generate a control signal based on $\overline{\text{HRESET_REQ}}$ and $\overline{\text{HRESET}}$ signals. This control signal could then be applied to a high speed active component (for example, FET) that is used on the MCKE signal to keep it low during assertion of $\overline{\text{HRESET}}$. 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 tDelay 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 components.

Option 4

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

Fix plan:

No plans to fix

DDR 9: DDR controller may not work with dual-ranked DDR3 registered DIMM if write leveling is enabled

Description: According to JEDEC specifications, the DDR controller must allow 3 DRAM cycles between back-to-back MRS (mode register set) commands when using DDR3 registered DIMMs. However, during write leveling, the DDR controller currently sends MRS commands on consecutive cycles (or on every other cycle for 2T timing mode). Therefore, the 3 DRAM cycles required between back-to-back MRS commands is not met in 1T or 2T timing mode.

Impact: DDR3 registered DIMMs with the write leveling may fail or yield invalid results. This could cause corrupt data to be written to DRAM during future write accesses.

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

- Use unbuffered DDR3 DIMMs.
- Use single-ranked DDR3 DIMMs.
- Disable write leveling if dual-ranked DDR3 registered DIMMs are used.
- Software Workaround #1 (for use with dual-ranked DDR3 registered DIMMs with write leveling enabled):

Note the following DEBUG register:

DEBUG_2 offset is CCSRBAR + DDR_OFFSET + 0xF04

Note The DRAM vendor should be consulted to determine whether the DIMMs operate properly with only two cycles between MRS commands.

- a. Configure DDR registers as is done in normal DDR configuration. Do not set DDR_SDRAM_CFG[MEM_EN].
- b. Clear DDR_SDRAM_CFG[RD_EN] to disable the registered DRAM setting to avoid conflict with 2T timing.
- c. Increment TIMING_CFG_1[CASLAT] to a value 1 cycle more than what is used by the DRAM.
- d. Increment TIMING_CFG_2[WR_LAT] to a value 1 cycle more than what is used by the DRAM.
- e. Set DDR_SDRAM_CFG[2T_EN] to enable 2T timing.
- f. Set DDR_SDRAM_CFG[MEM_EN].

Note: If it is preferred to use 1T timing for better performance, then DDR_SDRAM_CFG[2T_EN] may be cleared via the following sequence after the DDR controller has completed the initialization sequence (If DDR_SDRAM_CFG_2[D_INIT] was set prior to DDR_SDRAM_CFG[MEM_EN], D_INIT should be polled until it is cleared by hardware).

- a. Set DDR_SDRAM_CFG[MEM_HALT].
- b. Poll on DEBUG_2[30] until it is set by hardware.
- c. Set DDR_SDRAM_INTERVAL[REFINT] = 4'h0000 to Disable refreshes.
- d. Clear DDR_SDRAM_CFG[2T_EN].
- e. Set DDR_SDRAM_INTERVAL[REFINT] back to the original value.

If ECC is not enabled, go to f below. If ECC is enabled, then re-initialize the DRAM and go to the step below:

1. Set DDR_SDRAM_CFG_2[D_INIT] .
2. Poll on DDR_SDRAM_CFG_2[D_INIT] until it is cleared by hardware.
3. Wait calculated delay.

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

Delay = 400 ms/Gbytes x max for each memory controller memory size

For 32-bit data buses, multiply this number by 2.

Example: assume 64-bit DDR2, memory size = 1 Gbyte on memory controller 1.

Delay = 400 ms/GB x 1 Gbyte = 400 ms

f. Clear DDR_SDRAM_CFG[MEM_HALT].

- Software Workaround #2 (for use with dual-ranked DDR3 registered DIMMs with write leveling enabled):

Note: This workaround will not preserve the DDR contents. If the DRAM was placed in self-refresh to preserve DDR contents, this workaround should not be used.

Note the following DEBUG registers:

DEBUG_2 offset is CCSRBAR + DDR_OFFSET + 0xF04

DEBUG_3 offset is CCSRBAR + DDR_OFFSET + 0xF08

- a. Configure DDR registers as is done in normal DDR configuration. Do not set DDR_SDRAM_CFG[MEM_EN].
- b. Disable write leveling for DDR3 by clearing DDR_SDRAM_WRLVL_CNTL[WRLVL_EN].
- c. Disable automatic CAS to Preamble by setting TIMING_CFG_2[CPO] to any value other than 5b'11111.
- d. Disable refreshes by clearing DDR_SDRAM_INTERVAL[REFINT].
- e. Enable ZQ calibration if disabled previously.
- f. Set DDR_SDRAM_CFG[MEM_EN].
- g. Poll on DEBUG_2[30] until it is set by hardware.
- h. Clear DDR_SDRAM_CFG[MEM_EN].
- i. Disable Chip Select 1 if only one DIMM is used. Disable Chip Selects 1 and 3 if two DIMMs are used.
- j. Enable write leveling for DDR3 by setting DDR_SDRAM_WRLVL_CNTL[WRLVL_EN].
- k. Set DEBUG_3[21] to enable bit deskew, or start following the 8572 automatic CPO workaround if auto CPO is used.
- l. Enable refreshes by setting the correct refresh interval in DDR_SDRAM_INTERVAL[REFINT].
- m. Restore the DDR_SDRAM_MODE[ESDMODE] field such that the DRAM output buffers will be enabled.
- n. Set DDR_SDRAM_CFG[MEM_EN].
- o. Poll on DEBUG_2[30] until it is set by hardware.
- p. Enable Chip Select 1 if only one DIMM is used. Enable Chip Selects 1 and 3 if two DIMMs are used.
- q. Write to the DDR_SDRAM_MD_CNTL register to issue an MRS command to chip select 1 (and 3 if enabled). to enable to memory

devices' output drivers. On each write, use the following field values:

- MD_EN = 1'b1
 - CS_SEL set to the appropriate chip select
 - MD_SEL = 3'b001
 - SET_REF = 0'b0
 - SET_PRE = 0'b0
 - CKE_CNTL = 2'b00
 - MD_VALUE = same value that was programmed into DDR_SDRAM_MODE[ESDMODE]
- r. Throughout much of this sequence, refreshes were not issued to chip selects 1 and 3. However, no data was stored in memory that needed to be preserved. At this point, set DDR_SDRAM_CFG_2[D_INIT] to initialize data and ECC.
- s. Poll D_INIT to be cleared.

Fix plan: No plans to fix

DDR 10: DDR to CCB clock ratios of greater than 1.5:1 can result in DDR configuration register corruption

Description:	If the clock ratio between the DDR and CCB is greater than 1.5:1, during writes the DDR memory mapped registers may be corrupted and during reads incorrect data may be return by memory controller.
Impact:	DDR controller fails to function correctly.
Workaround:	The DDR to CCB clock ratio must be kept at or below 1.5:1. For example, if a DDR data rate of 800 MHz is used, then the minimum CCB frequency should be 533 MHz.
Fix plan:	Fixed in Rev 2.0

DDR 11: Asserting panic interrupt via `IRQ_OUT` the DDR controller may violate the JEDEC specifications

Description:	The DDR controller supports a panic interrupt that can be controlled by hardware. This interrupt can be used to place the DRAMs into self-refresh mode. However, the DDR controller can enter a bad state and violate JEDEC specifications for entering self-refresh when this interrupt is used.
Impact:	DRAM contents may become corrupted when self-refresh is entered. There may be other impacts related to the DRAM devices when the JEDEC specs are violated.
Workaround:	Instead of using the panic interrupt, the DDR controller can be placed into self-refresh without waiting on the part to enter a sleep state by asserting <code>DDR_SDRAM_CFG_2[FRC_SR]</code> .
Fix plan:	No plans to fix

DDR 12: The DDR controller may fail the write leveling sequence for DDR3

Description:	During write leveling, JEDEC allows the DDR3 SDRAM to drive status back to the DDR controller on either all DQ bits or the prime DQ bit. The prime DQ bit is typically defined as DQ[0] of each byte. The DDR controller currently uses DQ[0,8,16,24,32,40,48,56] and ECC[0] to read the status back for the various bytes of data. However, some DIMMs will be routed such that these data bits will not be connected to DQ[0] of each discrete device. Hence, if a particular vendor only drives status on a single bit during write leveling, this bit may get routed to a DQ bit that is different than what the DDR controller is expecting. Then, the controller will not observe the status correctly and will fail the write leveling sequence for DDR3.
Impact:	The write leveling sequence may fail, which could prevent the DDR controller from writing correctly to DRAM. This does not affect a system using DRAM discrete devices. It only affects a system using DDR3 DIMMs that drive the write leveling response on different data line than controller is expecting (that is, many DDR3 DIMM from various vendors will work, but others may not).
Workaround:	<p>Use one of the following options.</p> <ul style="list-style-type: none">• Use DDR3 memories that drive write leveling status on all DQ bits instead of a single DQ bit.• Modify the board design to connect DQ[0,8,16,24,32,40,48,56] and ECC[0] of the part to the bits carrying status on the DIMM connector. Depending upon the raw card design used, this may be different. This workaround limits the DIMM topologies that can be used for a particular system. For example, a single-ranked DIMM may have different routing for DQ[0] from each discrete than a dual-ranked DIMM.• Disable write leveling for DDR3 via DDR_SDRAM_WRLVL_CNTL[WRLVL_EN]. Instead, TIMING_CFG_2[WR_DATA_DLY] can be used to align DQS with MCK at the DRAM.
Fix plan:	No plans to fix

DDR 13: Change in DDRCLK input frequency range

Description:	The DDR controllers may be operated synchronously or asynchronously with respect to the CCB, depending upon Power-On Reset (POR) configuration (see Table 75, “DDR Clock Ratio,” in the MPC8572 Hardware Specifications). The range of the DDRCLK input clock – which is only required in asynchronous mode – is specified as 66-166MHz in Revision 1 of the device hardware specifications. However, if the DDRCLK input clock frequency is greater than 100MHz, DDR operation may be unreliable depending upon the external clock driver circuit being used.
Impact:	Systems using the DDR memory controllers in asynchronous mode with DDRCLK input frequency greater than 100 MHz may be impacted. System designs with DDRCLK input frequency less than or equal to 100 MHz are not affected by this erratum. Systems using the DDR memory controllers in synchronous mode are not affected by this erratum.
Workaround:	For asynchronous DDR, restrict DDRCLK input frequency to 66-100MHz.
Fix plan:	No plans to fix. The DDRCLK input frequency range will be modified to 66–100 MHz in Revision 2 of the device hardware specifications.

DDR 14: Data may become corrupted for DDR3 32-bit bus mode

Description:	When the DDR controller is operating in 32-bit bus mode using 8-beat bursts, it may corrupt the data during 32-byte accesses. The controller may incorrectly send the wrong data beats.
Impact:	No ECC error will be detected, as the error occurs past the ECC checking, and a full doubleword will be incorrect. Therefore, the incorrect data will be sent, and undefined results could occur. This only affects DDR3 mode.
Workaround:	For memory controller 1 write register at CCSRBAR Offset 0xE_0F34 with a value of 0x4000_0000 and for memory controller 2 write register at CCSRBAR Offset 0xE_0F3C with a value of 0x4000_0000, to allow DDR3 at 32-bit bus mode to operate properly.
Fix plan:	No plans to fix

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

Description:	As described in Table 19-3 in the MPC8572E reference manual (MPC8572ERM), the timing for $\overline{\text{DMA_DACKn}}$ output signal to the external DMA master is to be held for at least three system clocks. The DMA controller may violate this requirement, depending on the internal latency of a transaction. The DMA controller asserts $\overline{\text{DMA_DACKn}}$ based on an internal “write done” signal, which varies depending on the write target and whether it is the last write of a transaction. In most cases, the “write done” signal asserts too quickly, causing the $\overline{\text{DMA_DACKn}}$ to be held for too short a time.
Impact:	$\overline{\text{DMA_DACK}}$ does not meet minimum hold specification of 3 SYSCLK cycles.
Workaround:	None. In the external DMA master mode, the $\overline{\text{DMA_DACKn}}$ output is held for at least 16 CCB clock cycles.
Fix plan:	No plans to fix

DMA-A001: Uncorrectable data read errors for DMA transfers may also cause DMA lockup or additional detectable data corruption

Description:

If an uncorrectable ECC error, system bus timeout, or illegal address occurs in the system, an additional unexpected impact can cause occasional corruption of unrelated DMA streams (or, if multiple such errors occur, DMA lockup). Generally, it is expected that the measures used to deal with the initial read error mitigate any impact of this unexpected behavior.

DMA lockup is only realistic for programming errors and hard errors such as bad memories because it is not expected that the system will continue to operate without reset in the face of repeated uncorrectable read errors or timeouts, etc. Forwarding of corrupt data due to individual soft errors can be avoided as described in workaround option 2 below.

Freescale discovered this issue while debugging a test case in which intentionally injected programming errors caused a stream of illegal addresses that eventually resulted in a DMA controller hang.

The DMA controller can process up to 16 total read and write transactions at a time from up to 4 active channels. Under normal conditions, the reads return data which is then used to program the controller (descriptor read) or forwarded to the corresponding write address(es) (data read). If an error occurs on a read transaction, however, the controller may mis-handle the response, and end up corrupting the queue state in the DMA. This in turn may corrupt in-flight data streams (detectable). If multiple error responses occur, the DMA may lock up (channel stays busy forever and does not halt) and be unable to process new transactions.

Errors on reads can be from several sources, including the following:

- Illegal address (does not match valid LAW or CCSRBAR)
- Illegal target settings (for example, address maps to PCI-Express, but ATMU settings for that address are invalid)
- Target error (for example, multi-bit ECC error in DDR memory)
- End-to-end error (for example, error response packet on PCI-Express or Serial RapidIO)

Impact:

The impact of this behavior is expected to be low in the context of the hard system failure or data corruption or programming error that caused the behavior. That is, the measures taken to deal with the initial problem, such as taking the system offline (for system repair or software patch) or system reset (for soft errors) also take care of the DMA lockup or corrupted DMA transfer.

Below is a description of the possible additional impact of a failed DMA read.

An error on a DMA read transaction may corrupt the programmed state or data for a subsequent transfer from the same or a different channel.

A sequence of errors may cause DMA lockup in which the DMA is unable to process new transactions. In this case, at least one channel stays busy forever and never halts.

In all cases, SRn[TE] is set for channel n that originated the read transaction that had the error.

For DMA lockup, at least one channel, not necessarily the one that has SRn[TE] set, stays busy forever and does not halt.

Not every error causes or contributes to DMA corruption or lockup. A minimum of 4 errors is necessary to cause DMA lockup. A single error can cause data corruption if the DMA issues another transaction within a small window after an error response arrives for a prior transaction. Alignment and transfer size affect the possibility of new transactions falling in this window, as described in workarounds below.

Neither DMA lockup nor data corruption prevents software from being notified of the error condition via interrupt (nor does this erratum limit the ability of software to query error reporting registers or status registers).

Workaround:

The following steps can be taken to avoid any of the unexpected behavior caused by a DMA read data failure in case the steps taken to resolve the original data failure are not adequate to mitigate the additional unexpected behavior:

- Option 1: Limit DMA operation to 1 channel per DMA controller, and limit each descriptor to:
 - Byte count to ≤ 1 Kbyte. Byte count is set in BCRn[BC].
 - Aligned start addresses, such that each descriptor yields no more than 8 transfers (reads and writes, for example, 256 bytes aligned for a 1-Kbyte transfer yields 4 reads and 4 writes). Start addresses for reads are set in SADRn, and for writes in DADRn.
- Option 2: Limit DMA operation to 1 channel per DMA controller, and limit writes to 64 bytes aligned start addresses. In this scenario, DMA lockup is still possible, but not data corruption.
- Option 3: Ensure DMA reads do not get an error response by taking the following actions:
 - a. Validate that all DMA descriptor, source and destination addresses reference only valid LAW or CCSRBAR addresses, and valid target chip selects.
 - b. Do not host DMA descriptors on, or use the DMA engine to read from, external interfaces (for example, PCI-Express, serial RapidIO), which can return error responses. Writes to external interfaces are acceptable.
 - c. Disable ECC checking in the DDR controller by setting DDR_SDRAM_CFG[ECC_EN] = 0.
 - d. If hosting DMA descriptors on the eLBC, or using the DMA engine to read from the eLBC, then disable eLBC parity/ECC and timeout error checking by setting each of LTEDR[BMD, FCTD, PARD] = 1.
 - e. If L2 cache is enabled, or if hosting descriptors in L2 SRAM or reading from L2 as SRAM, then disable L2 parity/ECC checking by setting each of L2ERRDIS[TPARDIS, MBECCDIS] = 1.
- Option 3a: If detection of multi-bit ECC errors is not required in the system, then same as option 3 except for step c, which is replaced by the following:
 - c. Set ERR_DISABLE[MBED] = 1 and ERR_SBE[SBET] = 0xFF, and periodically poll and clear ERR_SBE[SBEC] to ensure it does not reach 0xFF.

NOTE

To switch from option 3 or 3a to option 1 or 2, in order to use the DMA controller to read from interfaces which may generate errors, software may halt the other DMA channels by setting $MRn[CS] = 0$ and wait for $SRn[CH] = 1$, then program the idle channel as in option 1 or option 2. When the transaction reading from the error-susceptible interface is complete ($MRn[CB] = 0$), resume multi-channel operation as in option 3 or 3a by setting $MRn[CS] = 1$ for the other 3 halted channels.

Fix plan: No plans to fix

eLBC 1: UPM does not have indication of completion of a Run Pattern special operation

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

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

The following two scenarios could be affected by this erratum:

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

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

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

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

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

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

Workaround: None

Fix plan: No plans to fix

eLBC 2: LTEATR and LTEAR may show incorrect values under certain scenarios

Description:	When FCM special operation is in progress, if any one of the errors/events that are listed in LTESR (except chip select error) occurs, the address and attribute that are captured in LTEAR and LTEATR may be incorrect.
Impact:	LTEAR and LTEATR cannot be used for debugging in this scenario.
Workaround:	None
Fix plan:	No plans to fix

eLBC 3: Some local bus events are not counted correctly in the performance monitor

Description:	The following events are not counted correctly: <ul style="list-style-type: none">• Bank 1–8 hits (chip-select)• Requests granted to ECM port• Cycles atomic reservation for ECM port is enabled
Impact:	These local bus events cannot be used in the performance monitor.
Workaround:	None
Fix plan:	No plans to fix

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

Description:	When the FCM is in the middle of a long transaction, such as NAND erase or write, another transaction on the GPCM or UPM triggers the bus monitor to start immediately for the GPCM or UPM, even though the GPCM or UPM is still waiting for the FCM to finish and has not yet started its transaction. If the bus monitor timeout value is not programmed for a sufficiently large value, the local bus monitor may time out. This timeout corrupts the current NAND Flash operation and terminate the GPCM or UPM operation.
Impact:	Local bus monitor may time out unexpectedly and corrupt the NAND transaction.
Workaround:	Set the local bus monitor timeout value to the maximum by setting LBCR[BMT] = 0 and LBCR[BMTPS] = 0xF.
Fix plan:	No plans to fix

eTSEC 1: Frame is dropped with collision and HALFDUP[Excess Defer] = 0

Description:	eTSEC drops excessively deferred frames without reporting error status when HALFDUP[Excess Defer] = 0. This erratum affects 10/100 Half Duplex modes only.
Impact:	The eTSEC does not correctly abort frames that are excessively deferred. Instead it closes the BD as if the frame is transmitted successfully. This results in the frame being dropped (because it is never transmitted) without any error status being reported to software.
Workaround:	Do not change HALFDUP[Excess Defer] from its default of 1. Thus eTSEC always tries to transmit frames regardless of the length of time the transmitter defers to carrier.
Fix plan:	No plans to fix

eTSEC 2: 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:	<p>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.</p> <p>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.</p> <p>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.</p>
Fix plan:	Fixed in Rev. 2.0.

eTSEC 3: Parsing of tunneled IP packets not supported

Description:	<p>Encapsulation of IP in IP in either TCP or UDP packets is not supported by eTSEC parser. This applies to both IPv4 and IPv6.</p> <p>A tunneled IP packet is an IP/TCP or IP/UDP packet and one of the following:</p> <ol style="list-style-type: none">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, respectively2. 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 <p>When the parser encounters a tunneled IP packet, it terminates its parsing operation at the end of the outer IP header.</p>
Impact:	<p>Because the parser terminates its parsing operation, only the outer IP header checksum, if it exists, is checked. Checksums of additional L3 headers within the packet are not checked, and no TCP/UDP checksums are checked.</p>
Workaround:	<p>If L3 or L4 parsing is enabled and tunneled packets are expected, software must examine each packet header to see if it is a tunneled IP packet. If the packet is a tunneled IP packet, software should calculate and check all checksums other than those in the outer IP header.</p>
Fix plan:	<p>Fixed in Rev. 2.0</p>

eTSEC 4: 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. 2.0.

eTSEC 5: 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 6 for more information.

Fix plan: Fixed in Rev. 1.1.1

eTSEC 6: 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 data word 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 be done by making sure that FCS at the end of the frame is incorrect, asserting TX_ER, or, in 16-bit FIFO encoded mode, sending an error code group. The controller does assert a local error event (IEVENT[DPE]), but does not give the link peer any error indication.

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

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

Workaround: None

Fix plan: Partially fixed in Rev 1.1.1, as follows:

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

1. Abort the transmitting packet with an error, all the applicable of:
 - a. truncated frame without CRC appended in FIFO modes
 - b. bad FCS in MAC modes
 - c. 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 IEVENT. 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 TBPTRn 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

Fixed in Rev. 2.0. as follows:

Same as above, but step 4 of software recovery routine not required. All other SW steps are part of normal error recovery sequence.

eTSEC 7: eTSEC half duplex receiver packet corruption

Description:	<p>When eTSEC is configured to run in half-duplex mode, it relies on the PHY to isolate its receiver from receive data during packet transmission. The PHY indicates contention on the wire via the COL signal in xMII, which forces the eTSEC into the standard backoff algorithm. If the eTSEC does in fact get receive data (RX_DV = 1) while it is transmitting (TX_EN = 1), it receives the packet data corrupted and inflected from its original size on its way to memory.</p> <p>This issue likely arises from the transmit data being wrapped at the PHY and send to the eTSEC receiver. This issue impacts running internal and external loopback while the eTSEC is configured for half-duplex operation.</p>
Impact:	<p>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.</p>
Workaround:	<p>The eTSEC can be configured through the filer to discard such packets that are wrapped externally in this manner through the use of MAC addresses match and drop. The receive MIB counters still count the packets as errored when they were in fact not errored.</p> <p>If loopback mode is desired, the eTSEC must be configured for full-duplex operation.</p>
Fix plan:	<p>Fixed in Rev. 1.1.1</p>

eTSEC 8: 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 Rx FIFO 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 Rx FIFO 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 PTV time. If the counter is already enabled, reset the count to 2/3 of PTV time. In the overflow event interrupt handler, check the current number of free receive buffer descriptors versus the threshold. If the number of free BDs are below the threshold, manually transmit a pause frame by setting `TCTRL[TFC_PAUSE] = 1`. This also restarts the lossless flow control state machine. In either case (free BDs above or below threshold), disable the counter until the next transmit control frame event.
- **Option 2:**
Enable interrupts on dropped packets due to lack of RxBDs (`EDIS[BSYDIS] = 0`, `IMASK[BSYEN] = 1`). On detecting a busy dropped packet event (`IEVENT[BSY] = 1`), manually transmit a pause frame by setting `TCTRL[TFC_PAUSE]` to restart the lossless flow control state machine.

NOTE

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

Fix plan:

Fixed in Rev. 2.0

eTSEC 9: 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. 2.0.

eTSEC 10: 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 of 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).

Some fetch errors are handled correctly. The correct behavior is as follows:

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.

The Tx reset sequence is:

- a. Set DMACTRL[GTSC]
- b. Poll IEVENT[GTSC] until set or 10,000 byte times elapse
- c. Clear MACCFG1[Tx_Flow]
- d. Wait 256 TX_CLK cycles
- e. Clear MACCFG1[Tx_EN]
- f. Wait 3 TX clocks
- g. Set MACCFG1[Reset Tx MC] and MACCFG1[Reset Tx fun]
- h. Wait 3 TX clocks
- i. Clear MACCFG1[Reset Tx MC] and MACCFG1[Reset Tx fun]
- j. Set TBPTRn to next available BD in the TX ring.
- k. Set MACCFG1[Tx EN] and, if desired, MACCFG1[Tx Flow]
- l. Clear DMACTRL[GTSC]

For Scenario 3, data errors can be flagged by platform for additional software processing, but no workaround exists to prevent the transmission of corrupted data.

Reference:

The information here is provided for reference purpose. Future MPC8572 Reference Manual will include the following description to clarify the eTSEC events that take place when fetch error condition occur:

- If EDIS[EBERRDIS] = 0,
 - the correct eTSEC response to a TxBD fetch error condition resulting from is to set IEVENT[EBERR] and halt all Tx queues with the error (TSTAT[THLTn] = 1, n = 0–7).
 - the correct eTSEC response to a RxBD fetch error condition is to set IEVENT[EBERR] and halt all Rx queues with the error (RSTAT[QHLTn] = 1, n = 0–7).

Fix plan:

Partially fixed in Rev 2.0

Scenarios 1 and 2 fixed

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

eTSEC 11: Rx TCP/UDP checksum checking may be incorrect while operating at low frequencies in FIFO mode

Description:	<p>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 ($RxFCB[ETU] = 0$).</p> <p>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).</p>
Impact:	<p>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.</p>
Workaround:	<p>Option 1: Turn off Rx checksum checking by setting $RCTRL[TUCSEN] = 0$.</p> <p>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).</p> <p>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)</p>
Fix plan:	<p>Fixed in Rev. 2.0.</p>

eTSEC 12: Parsing of MPLS label stack or non-IPv4/IPv6 label not supported

Description:	The parser does not continue parsing beyond multi-label stack, or MPLS frame with a label other than IPv4 or IPv6. The RxFCB is written as 0x0000_00ff_0000_0000 (no layer 3 header recognized).
Impact:	The eTSEC cannot parse beyond an MPLS stack of greater than depth 1. It also cannot parse beyond an MPLS header with label other than IPv4 or IPv6.
Workaround:	Limit MPLS Ether-type packets to MPLS label stack depth = 1 with IPv4 or IPv6 label.
Fix plan:	Fixed in Rev 2.0

eTSEC 13: Filer does not support matching against broadcast address flag PID1[EBC]

Description:	The controller clears its copy of the Ethernet broadcast address before extracting filer properties, so the filer cannot correctly match based on broadcast address (PID1[EBC]). The frame itself is not affected.
Impact:	If broadcast address matching is enabled, frames may be incorrectly filed or rejected.
Workaround:	Mask off matching on broadcast address flag (PID1[EBC] = 1) by clearing the bit 16 of the mask register. If the rule needs to be able to distinguish broadcast addresses as defined by IEEE Std 802.3™-2005 is all 1's in the destination address field, then use a filer rule with PID3 and PID4 (destination MAC address) to match on broadcast Ethernet frames.
Fix plan:	Fixed in Rev. 2.0.

eTSEC 14: eTSEC does not support parsing of LLC/SNAP/VLAN packets

Description:	eTSEC supports parsing of LLC/SNAP headers following the Ethernet 802.3 length field interpretation. It also supports 802.1p VLAN tags. However, it does not support the encapsulation defined as Ethernet length, followed by LLC/SNAP, followed by VLAN. The parser prematurely completes “normally” upon encountering the VLAN Ether-type at the end of the LLC/SNAP encoding. The erratum is that no layer 3, layer 4, or VLAN information is submitted to the filer or reported in the RxFCB.
Impact:	This unique packet type is not parsed beyond layer 2, including any VLAN processing, because the parser terminated before the VLAN tag was found.
Workaround:	Software running on the host has to parse these packets because they indicate no parsing functions performed by eTSEC.
Fix plan:	Fixed in Rev. 2.0.

eTSEC 15: eTSEC filer reports incorrect Ether-types with certain MPLS frames

Description:	The eTSEC filer gets a property under PID = 7 called ETY. This usually corresponds to the last Ether-type that was encountered in a packet as it was parsed. In the case that there is an MPLS label in the packet, then the Ether-type is incorrectly returned as the last 2 bytes of the MPLS label. The last 2 bytes correspond to the LSB 4 bits of the label, the EXP field, the S field, and the TTL field. The eTSEC does not know of any header types that follow other than IPv4 or IPv6 through the use of reserved label values "IPv4 Explicit Null" and "IPv6 Explicit Null." Hence the Ether-type is always left as the last 2 bytes of the MPLS label.
Impact:	MPLS tagged packets report the incorrect Ether-type (8847 for MPLS unicast or 8848 for MPLS multicast).
Workaround:	Use arbitrary extraction bytes to compare to the actual Ether-type if a filer rule is intending to file based on an MPLS label existence.
Fix plan:	Fixed in Rev. 2.0

eTSEC 16: 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. 2.0

eTSEC 17: 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. 1.1.1.

eTSEC 18: Parser does not check VER/TYPE of PPPoE packets

Description:	The Ethernet controller supports PPPoE VER/TYPE = 1 packets. For PPPoE packets with VER/TYPE not equal to 1, the controller should stop Ethernet parsing and treat it as an unrecognized PPPoE packet. Instead, the controller does not check the VER/TYPE field, and assumes it is 1.
Impact:	PPPoE packets with VER/TYPE that are not type 1 are parsed as if they are type 1 PPPoE.
Workaround:	None
Fix plan:	Fixed in Rev. 2.0.

eTSEC 19: 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. 1.1.1

eTSEC 20: 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 6 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. 2.0

eTSEC 21: No parser error for packets containing invalid IPv6 routing header packet

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

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

As part of the pseudo-header calculation for the L4 checksum, the controller uses the last destination address from the routing header. It then calculates the L4 checksum and leaves the ETU bit in the RxFCB clear (marking this as a good checksum.) Since the above conditions constitute an invalid IPv6 routing packet, the checksum should not be marked as good and a parse error should be flagged instead.

For Rev 1.1.1 and earlier, the logic would continue to parse the invalid packet normally as if there is no error in the packet. CTU and ETU bits work as if the packet is not invalid.

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

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

Impact: A packet with invalid IPv6 routing header will not be flagged as parse error. L4 checksum result (such as ETU) may be invalid.

Workaround: For Rev 1.1.1 and earlier:
Consistency checks for IPv6 routing header packets must be performed in software, or skipped. If a packet does have a valid IPv6 routing header, then L4 checksum result, if enabled, can be considered as valid. Otherwise, software should consider the packet as malformed and should not use the packet's L4 checksum result stored in RxFCB.

For Rev 2.0 and later:

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

Option 2 - Apply the same workaround as in Rev 1.1.1 only if CTU bit is 0.

Fix plan: Partially fixed in Rev 2.0

eTSEC 22: Generation of Ethernet pause frames may cause Tx lockup and false BD close

Description:	<p>The process of generating a PAUSE control frame may cause the controller to transmit two pause frames instead of one. If this occurs, the controller erroneously sees the second pause frame as a transmitted data frame, closes the next TxBD and decrements the running counter of frames in the TxFIFO.</p> <p>The controller can continue operating after this error, but with side-effects: a frame may still reside in the TxFIFO after its TxBD is closed, errors (for example, DPE) on a transmitting frame may be recorded in the wrong TxBD, and a frame in the TxFIFO is only transmitted if a second frame is also in the TxFIFO. This appears on the external interface as the controller being 'behind' on transmit, as the frame marked as transmitted in the closed TxBD is actually still in the FIFO waiting to transmit.</p> <p>These side-effects are cumulative: the second instance of a pause frame error causes the controller to be out of sync on TxBDs by two, and the TxFIFO requires three frames in the FIFO in order to transmit one. Ultimately, the controller locks up, as it runs out of room in the TxFIFO to hold enough frames to trigger transmit of the next frame.</p> <p>Once the controller "falls behind" or enters a Tx lockup state, only a reset of the eTSEC and associated software allows it to start transmitting normally again.</p> <p>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]).</p>
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:	<ul style="list-style-type: none">• 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.
Fix plan:	Fixed in Rev. 1.1.1

eTSEC 23: eTSEC parser does not perform length integrity checks

Description:	<p>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).</p> <p>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.</p>
Impact:	Could get false checksum failures or false checksum passes.
Workaround:	False checksum fails can be worked around by rechecking them in software running on the host.
Fix plan:	Fixed in Rev. 2.0.

eTSEC 24: eTSEC does not verify IPv6 routing header type field

Description:	The RFC2460 (current referenced standard for IPv6 operation) states that when encountering a packet with an unrecognized Routing Type Value, and the field "segments left" is non-zero, the node must discard the packet and return an ICMP Parameter problem, Code 0, message to the packet's source address. The eTSEC only recognizes type0 routing headers, but incorrectly interprets all Routing Type fields as type0 (for example, ignores the type field and continues parsing the packet including upper layer protocol checksums). The correct behavior is to signal parser error, and not check upper layer checksums
Impact:	eTSEC parser/checksum engine incorrectly interprets non-type0 IPv6 routing headers. Functionally, this is a future-proof issue because there are currently no other type-fields defined.
Workaround:	If this device is operating in a network that is using non-type0 IPv6 routing headers, then the upper layer processing (IPv6 extension headers, and payload checksumming operations) must be performed in software.
Fix plan:	Fixed in Rev. 2.0

eTSEC 25: Transmission of truncated frames may cause hang or lost data

Description:	<p>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:</p> <ol style="list-style-type: none">1. The Ethernet controller truncates a transmitted frame which is larger than MAXFRM2. The following frame has TOE = 13. The two frames together are large enough to fill the 10-Kbyte Tx FIFO without truncation
Impact:	<p>Truncating frames larger than MAXFRM may cause a transmit hang or lost data if combined with TOE = 1 frames.</p>
Workaround:	<ul style="list-style-type: none">• 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:	<p>Fixed in Rev. 2.0</p>

eTSEC 26: 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:	<ul style="list-style-type: none">• Option 1: Include a filter 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 9).
Fix plan:	Fixed in Rev. 2.0

eTSEC 27: Multiple BD frame may cause hang

Description:	<p>In the device reference manual, it states the following:</p> <p>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.</p> <p>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.</p>
Impact:	<p>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.</p>
Workaround:	<p>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.</p>
Fix plan:	<p>No plans to fix</p>

eTSEC 28: TxB[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 TxB[TC] = 1. In 16-bit FIFO modes (GMII-style or encoded), the TxB[TC] setting for a frame is not properly pipelined, so some frames with TxB[TC] = 1 end up transmitting without CRC appended and some frames with TxB[TC] = 0 end up with CRC appended by the controller.
Impact:	The Ethernet controller does not reliably append CRC based on TxB[TC] in 16-bit FIFO modes.
Workaround:	<ul style="list-style-type: none">• Option 1: Set FIFOCFG[CRCAPP] = 1 in 16-bit FIFO modes to force hardware append of CRC on all frames, regardless of TxB[TC] state.• Option 2: Set all TxB[TC] = 0.
Fix plan:	Fixed in Rev. 2.0.

eTSEC 29: Arbitrary Extraction Un-extractable Bytes

Description: The byte offset for the arbitrary extraction filter feature does not allow some byte offsets to 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 18–19 cannot be extracted.
- With 1 tag:
Packet bytes 22–23 cannot be extracted.
- With 2 tags:
Packet bytes 26–27 cannot be extracted.

Workaround: None.

Fix plan: Fixed in Rev. 1.1.1.

eTSEC 30: 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. 2.0

eTSEC 31: RxBD[TR] not asserted during truncation when last 4 bytes match CRC

Description:	The eTSEC truncates any receive frame larger than MAXFRM, unless Huge Frame Enable is set (MACCFG2[Huge Frame] = 1). The proper behavior for the controller is to set the RxBD[TR] bit (and RxBD[LG], if RxBD[L] = 1) for any truncated frame. If the last 4 data bytes received before truncation happens to match the running CRC, then RxBD[TR] (and RxBD[LG]) is not set even though the frame has been truncated.
Impact:	If the 4 data bytes just before MAXFRM bytes into the frame match the running CRC for the frame, the packet is silently truncated (no error indication via RxBD[TR]).
Workaround:	None
Fix plan:	Fixed in Rev. 2.0.

eTSEC 32: VLAN insertion and extraction cannot be used in FIFO mode

Description:	VLAN insertion and extraction were not implemented along with the rest of L2 parsing function in FIFO mode.
Impact:	VLAN insertion and extraction cannot be used in FIFO mode.
Workaround:	Set TCTRL[VLINS] = 0 and RCTRL[VLEX] = 0 in FIFO mode (ECTRL[FIFM = 1]).
Fix plan:	Fixed in Rev 2.0

eTSEC 33: 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 $RCTRL[PRSDPE] = 11$, $RCTRL[TUCSEN] = 1$):

- For IP total length of $4n + 1$, the L4 byte offsets $4n + m - \langle \text{IP header length} \rangle$ are not extractable, for $m = 1, 2$, or 3 .
- For IP total length of $4n + 2$, the L4 byte offsets $4n + m - \langle \text{IP header length} \rangle$ are not extractable, for $m = 2$ or 3 .
- For IP total length of $4n + 3$, the L4 byte offset $4n + 3 - \langle \text{IP header length} \rangle$ is not extractable

Workaround: None

Fix plan: Fixed in Rev. 2.0.

eTSEC 34: L2 arb extract shifted with shim headers not multiple of 4 bytes

Description:	L2 arbitrary extraction offsets ≥ 8 ordinarily start at the DA of the L2 header. If shim headers are enabled ($RCTRL[L2OFF] \neq 0$), L2 arb extract offsets ≥ 8 start at the beginning of the shim header, and continue into the regular L2 header if the shim header is less than 56 bytes long. If the shim header length is not a multiple of 4 bytes, the controller does not take the byte shift into account when extracting from the regular L2 header and extracts the wrong bytes.
Impact:	L2 arbitrary extraction is not supported if shim headers are enabled and they are not a multiple of 4 bytes in length.
Workaround:	<ul style="list-style-type: none">• Option 1: Do not use shim headers ($RCTRL[L2OFF] = 0$).• Option 2: Set the shim header length to multiples of 4 bytes.
Fix plan:	Fixed in Rev. 2.0.

eTSEC 35: 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[IPCSSEN] 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. 2.0.

eTSEC 36: Frames greater than 9600 bytes with TOE = 1 will hang controller

Description:	The eTSEC supports frames up to 9600 bytes (huge or jumbo frame). If a frame has TOE = 1, it must be no more than 9600 bytes to fit entirely into the Tx FIFO. If the frame is larger, the controller hangs, because it must have the last byte of data in the FIFO to calculate the checksum and allow the frame to start transmission.
Impact:	A frame larger than 9600 bytes with TOE = 1 hangs the Ethernet controller.
Workaround:	For frames larger than 9600 bytes, set TxBD[TOE] = 0. For frames with TxBD[TOE] = 1, ensure Tx frame length <= 9600 bytes.
Fix plan:	Fixed in Rev. 2.0.

eTSEC 37: Setting RCTRL[LFC] = 0 may not immediately disable LFC

Description:	Lossless flow control is controlled by RCTRL[LFC]. Setting RCTRL[LFC] = 0 should immediately disable the lossless flow control state machine and stop the sending of pause frames based on number of free RxBDs. The controller instead waits until the state machine is idle before disabling it. If the state machine has been triggered by the number of free RxBDs falling below the threshold, the controller continues sending pause frame extensions until the number of free RxBDs exceeds the threshold.
Impact:	Generation of pause frames due to lack of free RxBDs may continue for a time after setting RCTRL[LFC] = 0.
Workaround:	When disabling LFC, first set RCTRL[LFC] = 0, then poll the number of free RxBDs until it exceeds the threshold. Once the number of free RxBDs exceeds the threshold, the configuration for LFC may be safely modified.
Fix plan:	Fixed in Rev. 2.0.

eTSEC 38: False parity error at Tx startup

Description:	The 10 KB TxFIFO comes out of reset in an uninitialized 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:	<p>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.</p> <p>Rev 1.1.1 only:</p> <p>Also, the false parity error may cause FCS corruption on the transmitting frame, if there is one.</p>
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. 2.0

eTSEC 39: 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 DA (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: <ul style="list-style-type: none">• 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 40: Transmit fails to utilize 100% of line bandwidth

Description:	<p>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.</p> <ul style="list-style-type: none">• For single-queue operation, IPG varies between 12–20 cycles.• For multiple queue operation with fixed priority scheduling, IPG for back-to-back frames from different queues varies between 70–140 cycles for a 2:1 eTSEC system clock:TSECn_TX_CLK ratio and twice as many cycles for a 1:1 ratio.• 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. <p>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.</p>
Impact:	<p>Tx bandwidth cannot achieve 100% line rate, especially for multiple queue operation or relatively small frames.</p>
Workaround:	<p>The following options maximize the bandwidth utilized by the Ethernet controller.</p> <ul style="list-style-type: none">• 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:	<p>Partially fixed in Rev 2.0.</p> <p>Partial fix as follows:</p> <p>Fixed for single-queue operation, with a single BD per frame. Under these conditions, the eTSEC will adhere to the 12-cycle IPG, allowing 100% line bandwidth to be reached.</p> <p>The multiple-queue recommendations provided in the workaround above still apply for silicon version 2.0.</p>

eTSEC 41: eTSEC4 ECNTRL[GMIIM] bit does not set in RGMII mode

Description:	eTSEC4 ECNTRL[GMIIM] bit does not get set when the interface is configured for RGMII mode.
Impact:	No functional impact. The only impact is that software cannot rely on ECNTRL[GMIIM] bit to determine whether the interface is configured for RGMII mode or not.
Workaround:	Software need to decode PORDEVSR[ECP4] to determine eTSEC4 controller protocol.
Fix plan:	Fixed in Rev. 2.0.

eTSEC 42: 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 IPv6 frames, not IPv4.
Impact:	False ETU indication when check sum for pseudo header SA is 0x1_0000 for IPv6 frames.
Workaround:	If RxFCB[CTU]=1, RxFCB[ETU]=1 and RxFCB[IP6]=1, calculate the checksum for the SA field from the pseudo header. If this checksum equals 0x1_0000, then proceed to calculate the entire TCP checksum to be sure the checksum error is valid. If the SA checksum is not 0x1_0000, then the ETU is a valid checksum error indication.
Fix plan:	Fixed in Rev. 2.0

eTSEC 43: 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: <ul style="list-style-type: none">• Disable user-defined Tx preamble by setting MACCFG2[PreAmTxEn] = 0.• Disable IP and TCP/UDP checksum generation by setting TCTRL[IPCSSEN]=0 and TCTRL[TUCSEN] = 0.
Fix plan:	No plans to fix

eTSEC 44: 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:	Fixed in Rev. 2.0.

eTSEC 45: Rx packet padding limitations at low clock ratios

Description:	<p>There are two mechanisms that cause extra bytes to be inserted in front of the data in a received frame:</p> <ol style="list-style-type: none">1. RCTRL[PAL] - packet alignment padding. A programmable mechanism for padding a frame with zeroes to achieve a particular alignment of data. Additionally if the 1588 time-stamping feature is enabled, the padding includes the 8 bytes of 1588 Rx timestamp data.2. MACCFG2[PreamRxEn] – enables inserting the 8-byte preamble in front of the Rx frame data within the data buffer. These bytes are not accounted for in the value of RCTRL[PAL] setting. <p>At low clock ratios (less than 4:1 platform clock to MAC interface clock ratio), it is possible for the receive buffer to overflow when 24 or more extra bytes are inserted into the Rx data buffer. When this Rx buffer overflow occurs, the current Rx frame is dropped and the subsequent frame may be passed to memory without the expected padding bytes inserted.</p> <p>The MAC interface clock rate is determined by the interface configuration and link data rate. Parallel MAC interfaces operating at 1000 Mbps, such as GMII and TBI, have a 125 MHz MAC interface clock rate. Parallel interfaces operating at either 100 Mbps or 10 Mbps have a MAC interface clock rate that is ¼ of the bit rate (25 MHz at 100 Mbps and 2.5 MHz at 10 Mbps). When operating in SGMII mode, the MAC interface clock rate is determined by the actual link data rate, exclusive of frame elongation bytes that are used when transferring 10 Mbps and 100 Mbps data across the link. In SGMII mode, the MAC interface clock rate is 125 MHz for 1000 Mbps data rate, 25 MHz for 100 Mbps data rate, and 2.5 MHz for 10 MHz data rate.</p>
Impact:	<p>If the platform clock is less than 500 MHz and the eTSEC is operating at a 1000 Mbps data rate (regardless of interface configuration), the eTSEC cannot support inserting 24 or more total bytes (from padding, time-stamping and the preamble) in front of the Rx frame data.</p>
Workaround:	<ul style="list-style-type: none">• Limit total receive packet byte insertion via RCTRL[PAL], 1588 time-stamping, and Rx preamble enable to less than 24 bytes total when the platform clock is less than 500 MHz and the interface is operating at 1000 Mbps data rate.• Limit the eTSEC data rate to 100 Mbps or less when the platform clock is less than 500 MHz. This can be accomplished by using an RGMII, MII, RMII, or SGMII interface in conjunction with setting MACCFG2[I/F Mode]=01.
Fix plan:	<p>Fixed in Rev. 2.0.</p>

eTSEC 46: Controller stops transmitting pause control frames

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

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

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

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

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

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

Options to minimize the frequency of the error condition:

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

1. Minimize or avoid the use of software-generated pause control frames.
Pause control frames are generated by software by setting TCTRL[TFC_PAUSE] = 1. Note that if the pause control state machine is stuck, neither IEVENT[GTSC] nor IEVENT[TXC] will be set to 1 as a result of setting TCTRL[TFC_PAUSE] = 1, and TFC_PAUSE will never be reset to 0. Transmission of data frames will continue while TFC_PAUSE stays stuck at 1.
2. Do not use lossless flow control. Lossless flow control is enabled by setting RCTRL[LFC] = 1.
3. If using lossless flow control, increase the value of PTV such that two LFC-triggered pause frames cannot occur within the failing window.

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

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

Detection and recovery:

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

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

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

Fix plan:

Fixed in Rev. 2.0.

eTSEC 47: 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 OVRFLW, which is detected through the CARN
register.
{ACCUM_HI_CO,ACCUM_HI} = {1'b0,ACCUM_HI} + ACCUM_LO_CO + OVRFLW;
```

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

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

Fix plan: No plans to fix

eTSEC 48: 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. 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 49: 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:	<p>Option 1:</p> <p>Set MACCFG2[PAD/CRC] = 1, which pads all short Tx frames to 64 bytes.</p> <p>Option 2:</p> <p>Use software-generated CRC (MACCFG2[PAD/CRC] = 0, MACCFG2[CRC EN] = 0 and TxBD[TC] = 0)</p>
Fix plan:	No plans to fix

eTSEC 50: 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_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_<complete sequence>

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 2.0
Sequences a) and b)—fixed
Sequence c)—no plans to fix

eTSEC 51: MAC: Malformed Magic Packet Triggers Magic Packet Exit

Description:	<p>The Ethernet MAC should recognize Magic Packet sequences as follows:</p> <p>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.</p> <p>Once the Ethernet MAC has recognized a valid DA for one frame, it continues searching for valid 102-byte Magic Packet sequences through multiple frames without checking for valid DA on each frame. The only events that cause the MAC to go back to check for valid DA before checking for a Magic Packet sequence on new frames are:</p> <ol style="list-style-type: none">1. A frame containing a recognized full Magic Packet sequence (with valid or invalid FCS)2. Software disable of Magic Packet mode (MACCFG2[MPEN]=0)3. MAC soft reset (MACCFG1[Soft_Reset]=1)
Impact:	<p>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.</p>
Workaround:	<p>None</p>
Fix plan:	<p>Fixed in Rev. 2.0</p>

eTSEC 52: Receive pause frame with PTV = 0 does not resume transmission

Description:	The Ethernet controller supports receive flow control using pause frames. If a pause frame is received, the controller sets a pause time counter to the control frame's pause time value, and stops transmitting frames as long as the counter is non-zero. The counter decrements once for every 512 bit-times. If a pause frame is received while the transmitter is still in pause state, the control frame's pause time value replaces the current value of the pause time counter, with the special case that if the pause control frame's pause time value is 0, the transmitter should exit pause state immediately. The controller does use the frame's pause time value to set the current pause time counter, but it then decrements the pause time counter before performing the compare to zero. As a result an XON (pause frame with PTV = 0), actually causes the transmitter to continue in pause state for 65,535 pause quanta, or 33,553,920 bit times.
Impact:	A received pause frame with PTV = 0 causes the transmitter to pause for 65,535 pause_quanta. The expected behavior is for the controller to continue, or resume, transmission immediately. Note that the Ethernet controller always uses the value of the PTV register when generating pause frames. It never automatically generates a pause frame with pause time value of 0 when the receiver recovers from being above the RxFIFO threshold or below the free RxBDs threshold.
Workaround:	To force an exit of pause state, use a pause frame with PTV value of 1 instead of 0.
Fix plan:	Fixed in Rev. 2.0

eTSEC 53: 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:	<p>To reduce the probability of an RxFIFO overflow, enable flow control by setting MACCFG1[Tx Flow] = 1.</p> <p>Statistical lockup detection and recovery:</p> <p>Lockup detection:</p> <ol style="list-style-type: none">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. <p>Lockup recovery:</p> <ol style="list-style-type: none">1. Perform a graceful receive stop by setting DMACTL[GRS] = 1, and wait to ensure any outstanding prefetches are cleared. The wait time is system and memory dependent, but a reasonable worst-case time is the receive time for a 9.6 KB frame at 10/100/1000 Mbps). Note that if the Rx is truly locked up, IEVENT[GRSC] will never be set. The graceful receive stop also ensures that data and state are not corrupted during a soft reset if the lockup detection falsely detects a lockup due to rejected packets.2. Toggle MACCFG1[Rx En] (set to 0, then set to 1).3. Clear the graceful receive stop by setting DMACTL[GRS] = 0.
Fix plan:	Fixed in Rev. 2.0

eTSEC 54: TxBD polling loop latency is 1024 bit-times instead of 512

Description:	Register bit DMACTRL[WOP] defines the use of wait on poll when transmit ring scheduling algorithm is set to single polled ring mode. (TCTRL[TXSCHED]=00). When the use polling is selected by setting DMACTRL[WOP]=0, the poll to TxBD on ring 0 should occur every 512 bit-times. Due to the errata the poll occurs every 1024 bit-times.
Impact:	The duration of the polling is twice as long as originally specified.
Workaround:	None
Fix plan:	No plans to fix

eTSEC 55: Misfiled Packets Due to Incorrect Rx Filer Set Mask Rollback

Description:

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

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

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

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

Rule #2: <Mask = 0x0000_8000>

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

Rule #3: <Mask = 0x0000_8000>

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

Rule #4: <Mask = 0x0000_8000>

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

Rule #5: <Mask = 0x0000_0210>

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

Rule #6: <Mask = 0x0000_8000>

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

Rule #7: <Mask = 0xFF00_FFFF>

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

Rule #8: <Mask = 0xFF00_FFFF>

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

Rule #9: <Mask = 0xFF00_FFFF>

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

Rule #10: <Mask = 0xFF00_FFFF>

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

Rule #11: <Mask = 0xFF00_FFFF>

etc.

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

Rule #8: <Mask = 0xFF00_FFFF>

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

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

Rule #9: <Mask = 0x0000_8000>

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

Rule #10: <Mask = 0x0000_8000>

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

Rule #11: <Mask = 0x0000_8000>

Mask should be 0xFF00_FFFF.

etc.

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

Impact:

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

Workaround:

Use one of the following workarounds:

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

Fix plan:

No plans to fix

eTSEC 56: Excess delays when transmitting TOE=1 large frames

Description:	<p>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[IPCSSEN]=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.</p> <p>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.</p>
Impact:	TOE=1 frames larger than 2500 bytes may see excess delays before start of transmission.
Workaround:	<p>Limit TOE=1 frames to less than 2500 bytes to avoid excess delays due to memory throttling.</p> <p>When using packets larger than 2700 bytes, it is recommended to turn TOE off.</p>
Fix plan:	No plans to fix

eTSEC 57: Data corruption may occur in SGMII mode

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

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

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

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

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

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

After these events, normal data reception resumes.

Workaround: Use one of the following options:

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

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

Fix plan: Plan to fix in Rev 2.1

eTSEC 58: Incorrect frame data in L3+L4-only or L4-only parsing for FIFO mode

Description:	<p>eTSEC supports a variety of parsing options in FIFO mode (ECNTRL[FIFM]=1) based on the setting of RCTRL[PRSDP] and RCTRL[PRFSM]. If PRSDP=10, then PRFSM determines whether parsing is of L2+L3, or L3 only. If PRSDP=11, then PRFSM determines whether parsing is of L2-L4, or L3-L4 only.</p> <p>When PRFSM=0, the parser is supposed to skip all L2 parsing functions and assume the frame starts with an L3 header. If the 13th and 14th bytes of the L3 header, corresponding to the byte positions for the type field in an L2 header, happen to match the L2 type values for a control frame (0x8808), however, and RCTRL[CFA]=0, then the L2 parsing logic will corrupt some of the frame data sent to memory. Similarly, if the 13th and 14th bytes of the L3 header happen to match the VLAN L2 type (0x8100 or DFVLAN[Tag]), and RCTRL[VLEX]=1, then the L2 parsing logic will corrupt some of the frame data sent to memory.</p>
Impact:	<p>When L3-only or L3+L4 header parsing is enabled in FIFO mode, the frame may be corrupted in memory if it appears to match the ethertype of a control or VLAN frame.</p>
Workaround:	<p>Set RCTRL[CFA]=1 and RCTRL[VLEX]=0.</p> <p>Setting CFA=1 prevents the eTSEC from corrupting a frame that appears to match a control frame ethertype (0x8808).</p> <p>Setting VLEX=0 prevents the eTSEC from corrupting a frame that appears to match a VLAN ethertype (0x8100 or DFVLAN[Tag]).</p>
Fix plan:	<p>No plans to fix</p>

eTSEC 59: 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:	<p>The Ethernet controller may be unable to transmit a pause frame during pause state if a normal frame is ready to transmit.</p> <p>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]).</p>
Workaround:	None
Fix plan:	No plans to fix

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

Description:	When the Ethernet controller receives a pause frame with PTV!=0, and MACCFG1[Rx Flow]=1, it completes transmitting any current frame in progress, then should pause for PTV*512 bit times. The MAC, however, does not take the full transmission time of the current frame into account when calculating the Tx pause time, and may pause for 1-2 pause quanta (512-1024 bit times) less than the PTV value.
Impact:	<p>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.</p> <p>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.</p>
Workaround:	Add 2 pause quanta to the pause time value used when generating pause frames to prevent receive buffer overflow.
Fix plan:	No plans to fix

eTSEC-A002: GRS may fail to complete after receiving a 1- or 2-byte frame

Description:	<p>The Ethernet controller provides a mechanism to quiesce the receive function via a graceful receive stop (DMACTRL[GRS]) request. When GRS is asserted, the controller rejects all new frames, waits for the receive FIFO to finish transferring all data and buffer descriptor updates to memory, then signals that the graceful receive stop has completed by setting IEVENT[GRSC].</p> <p>If the controller receives a 1- or 2-byte frame (such as an illegal runt packet or a packet with RX_ER asserted) before GRS is asserted and does not receive any other frames, the controller may fail to set GRSC even when the receive logic is completely idle. Any subsequent receive frame that is larger than two bytes will reset the state so the graceful stop can complete. A MAC receiver (Rx) reset will also reset the state.</p>
Impact:	<p>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.</p>
Workaround:	<p>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.</p> <p>MAX Rx reset procedure:</p> <ol style="list-style-type: none">1) Clear MACCFG[RX_EN].2) Wait three Rx clocks.3) Set MACCFG2[RX_EN].
Fix plan:	<p>No plans to fix</p>

IEEE1588_1: 1588 reference clock limited to 1/2 controller core clock in asynchronous mode

Description:	If the eTSEC system clock is not at least twice the 1588 reference clock frequency, the PTP ID from TxFCB[VLCTL] may not be captured correctly in the TMR_TXTS1/2_ID (transmit time stamp identification) register.
Impact:	A fast 1588 reference clock may lead to an invalid PTP ID in the TMR_TXTS1/2_ID register.
Workaround:	Run the 1588 reference clock synchronously (TMR_CTRL[CKSEL] = 01). Refer to IEEE 1588_6.
Fix plan:	Fixed in Rev 2.0

IEEE1588_2: IEEE 1588 not supported in SGMII mode

Description:	The eTSEC controller does not properly support the time-stamping of packets on the SGMII interface.
Impact:	IEEE 1588 functionality is not supported in SGMII mode.
Workaround:	None.
Fix plan:	Fixed in Rev 1.1.1.

IEEE1588_3: 1588 reference clock pulse required between writes to TMR_ALARMn_L and TMR_ALARMn_H

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

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

Workaround:

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

Fix plan: Fixed in Rev 2.0

IEEE1588_4: 1588 alarm fires when programmed to less than current time

Description:	The 1588 alarm mechanism operates purely on a less-than-or-equal-to comparison with the current time. If the alarm is programmed to a value less than the current time, it fires immediately, rather than waiting for the current time to wrap around and cross the alarm time.
Impact:	Alarm may fire before the intended time if armed when current time value is greater than alarm value.
Workaround:	Ensure that the current time is less than the intended alarm time when programming TMR_ALARMn_H to arm the event.
Fix plan:	Partially fixed in Rev 2.0 Timer alarm 1- Fixed in Rev 2.0. Timer alarm 2 - No plans to fix

IEEE1588_5: IEEE 1588 not supported in Dual-Clock TBI mode

Description:	The eTSEC controller should support 1588 functionality in all Ethernet modes, including Dual-Clock TBI. However in this mode, the controller selects the wrong clocking source which prevents the proper 1588 operation.
Impact:	IEEE 1588 functionality is not supported in Dual-Clock TBI mode.
Workaround:	None.
Fix plan:	Fixed in Rev 2.0

IEEE1588_6: Use of asynchronous 1588 reference clock may cause errors

Description:	There is 1588 clock-domain information related to the timestamp that must cross to the eTSEC system clock domain and is not properly synchronized. During 1588 negotiation, S/W must also write some of the 1588 registers. The new register values then cross from the eTSEC system clock domain to the 1588 clock domain where the 1588 state machines reside. That crossing is also unsynchronized, with the effect that the register values are unstable for a few cycles and may cause improper operation of the state machines.
Impact:	If the 1588 clock and the eTSEC system clock are asynchronous, then there is no guarantee that all bits in the 1588 registers and state machines are stable when they are latched by the eTSEC system clock. This can lead to cases where counter values “appear” to jump forward or run backwards (due to the settling of the bits), or where state machines transition improperly to new states.
Workaround:	This asynchronous boundary problem can be avoided by clocking the 1588 clock domain using the eTSEC system clock, thus guaranteeing that the two domains are synchronous with each other. eTSEC system clock is selected as the 1588 reference clock by setting TMR_CTRL[CKSEL] = b01.
Fix plan:	Fixed in Rev 2.0.

IEEE1588_7: Cannot use inverted 1588 reference clock when selecting eTSEC system clock as source

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

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

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

Workaround: None

Fix plan: Fixed in Rev 2.0

IEEE1588_8: Odd prescale values not supported

Description:	<p>The 1588 timer prescale register (TMR_PRSC) defines the timer prescale as follows: PRSC_OCK: Output clock division/prescale factor. Output clock is generated by dividing the timer input clock by this number. Programmed value in this field must be greater than 1. Any value less than 1 is treated as 2.</p> <p>The output pulse (TSEC_TMR_PPN) width should be 1x the output clock width. For odd prescale values, the pulse width is 1.5x the output clock width instead.</p>
Impact:	Odd 1588 timer prescale values are not supported.
Workaround:	Use only even timer prescale values.
Fix plan:	No plans to fix

IEEE1588_9: FIPER's periodic pulse phase not realigned when the 1588 current time is adjusted

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

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

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

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

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

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

Note that there will be no FIPER pulses between steps 1 and 4 for the first sequence and between steps 1 and 3 of the second sequence.

Fix plan: Fixed in Rev 2.0

IEEE1588_UM1: TMR_ALARMn_L register must be written before TMR_ALARMn_H

Description: The 1588 timer alarm register is a 64-bit entity implemented as two 32-bit registers. The low register must be written before the high register for proper function. If written in the reverse order, the alarm does not transition to the ARMED state.

The current reference manual does not mention this requirement, but says:

“Alarm time comparator register. Comparator output goes high when the current time counter becomes equal to or greater than the alarm time comparator value. User reprograms the TMR_ALARMn_H/L registers to deactivate their corresponding outputs.”

Impact: The alarm does not transition to the ARMED state, if the above high register is written before the low register.

Workaround: None.

Fix plan: Reference manual update

The reference manual will have this description updated regarding the “Alarm time comparator register”:

Alarm time comparator register. The corresponding alarm event in TMR_TEVENT is set when the current time counter becomes equal to or greater than the alarm time compare value in TMR_ALARMn_L/H. Writing the TMR_ALARMn_L register deactivates the alarm event after it has fired. Writing the TMR_ALARMn_L followed by the TMR_ALARMn_H register rearms the alarm function with the new compare value.

IEEE1588_10: IEEE 1588 accuracy can be adversely impacted in systems using multiple unsynchronized gigabit Ethernet ports

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

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

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

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

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

Fix plan: No plans to fix

IEEE1588-A001: Missing or incorrect received frame timestamp values occur when 1588 time-stamping is enabled

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

Impact: The specific impact of this erratum is dependent on both interface mode configuration and link rate, as listed below:

Table 5.

Interface Configuration	Link Rate	Erratum Impact
GMII/RGMII	1 Gbps	Recorded frame timestamp may be incorrect
GMII/MII/RGMII/SGMII	10/100 Mbps	Recorded frame timestamp may be incorrect Received frame may be dropped
TBI/RTBI/SGMII	1 Gbps	Recorded frame timestamp may be incorrect Received frame may be dropped

This erratum does not affect the following:

- Frames received on ports configured as RMII operating at a 10/100Mbps link rate
- The operation of the TRIGGER_IN inputs, ALARM outputs, or FIPER outputs
- The operation of an Ethernet port when time-stamping is disabled (TMR_CTRL[TE] = 0 and RCTRL[TS] = 0); HRESET default is disabled

Workaround:

- If receive frame time-stamping is required on a particular port, operate the port in RMII 10/100 Mbps mode.
- If receive frame time-stamping is not required on a particular port, disable the frame time-stamping operation on the port by clearing RCTRL[TS].

Fix plan: No plans to fix

FEC 1: MII, half duplex collision causes Tx frames to lose data

Description: A problem exists in the Ethernet controller in which frame data can be lost when running in half duplex mode and transferring frames that exceed the size of the FIFO (1 Kbyte). The problem manifests itself when a frame that exceeds the FIFO size gets retried multiple times. In such case, there is a possibility that the FIFO read and write pointers fall out sync resulting in frame data being lost.

A watermark indicator gets communicated from the FIFO to the DMA telling it that there is space available in the FIFO to write new data. In the default case, the FIFO asserts this indicator when it believes it has 64 bytes available in the FIFO. This indicator works fine when there is no retry. However, when a retry occurs within the collision window, and the FIFO is full, the FIFO may falsely indicate to the DMA that it has space available. Upon getting a retry, the FIFO resets the FIFO pointer back to the beginning of the frame to begin re-transmission, but at that point the DMA may have written new data into an already full FIFO causing the FIFO write and read pointers to get out of sync, and resulting in a loss of frame data.

Impact: Frames may be corrupted when the Ethernet controller is run in half-duplex mode and a collision occurs while transferring a frame that exceeds the FIFO size. With this erratum, the first 1024 bytes are lost.

Workaround: Program the FEC's respective debug register with a value of 0x25. The FEC register is located at offset 0x2_8094.

This register is a debug register whose content is used to compare with the number of free entries left in the Tx FIFO. Once the number of free entries is less than what is programmed in this debug register, the DMA is then only allowed to write 64 more bytes of data. The DMA can only resume writing after the number of free space in the Tx FIFO exceeds what is programmed in the debug register. The value 0x25 is calculated from adding an additional 64-bytes of frame data plus 20 bytes of elastic FIFO used for boundary crossing logic to the default 64-byte value:

$64 + 20 + 64 = 148$ bytes or 0x25 entries

This value ensures that the DMA engine is held-off until the collision window is passed before writing more data into the Tx FIFO

Fix plan: No plans to fix

FEC 2: FEC: Rx_FIFO overflow due to system busy may cause receiver to hang

Description: It is possible for the FEC's Rx FIFO to encounter an overflow situation due to a system busy condition caused by heavy accesses to DDR memory from competing devices and FEC. It may take longer for the FEC to access DDR memory, therefore the possibility of an overflow condition is increased especially due to heavy line load and at Gigabit rate. Note that a system busy condition is not due to a lack of Rx buffers as indicated by IEVENT[BSY] bit or a GRS condition as indicated by IEVENT[GRSC] bit. The symptom is the receiver may drop frames but with no dropped frame indication. In some cases, the receiver may hang. When the receiver enters a hung state, all incoming frames are dropped and the RDRP register logs these as dropped frames. Only a hard reset to the chip gets the FEC out of this condition.

Impact: Receiver may hang under certain failed scenarios or there are lost frames but with no dropped frame indication. Typically, this may happen when FEC is operating under the condition of small packets (for example, 64-byte) with heavy line load.

Workaround:

1. Software needs to put Tx and Rx buffer descriptors in L2 cache. This can be achieved by performing a DCBTL (Touch and Lock Set) instruction to lock the Tx and Rx buffer descriptors in L2 cache in the beginning. Then turn on Rx buffer descriptor stashing by programming the ATTR[BDLWT] = 10.
2. Perform stashing on the first 64-byte of frame data. This can be done by programming ATTR[ELCWT] = 10, ATTRELI[EL] = 0x40, ATTRELI[EI] = 0.
3. Put the FEC in interrupt coalescing mode with the ICFCT field of the TXIC and RXIC set to something other than 1 (for example, 5).
4. Change the alarm and panic watermark levels to a lower level if this problem is experienced. These watermarks are currently resided in debug registers. Changing the watermark level to a lower level increases the priority of the Rx FIFO's flushing requests to memory earlier and a higher share of the memory bandwidth. The priority is raised higher when the amount of data in the Rx FIFO pass these watermark levels, therefore ensuring that they have higher priority in flushing the packet's data currently in the Rx FIFO to memory. These registers are FIFO_RX_ALARM, FIFO_RX_ALARM_SHUTOFF, FIFO_RX_PANIC, and FIFO_RX_PANIC_SHUTOFF.

For FEC:

FIFO_RX_ALARM, at offset 0x050, is by default set at 0x080 (1/2 full).

New recommended value would be at 0x002.


FIFO_RX_ALARM_SHUTOFF, at offset 0x054, is by default set at 0x040 (1/4 full).

New recommended value would be at 0x001.

FIFO_RX_PANIC, at offset 0x058, is by default set at 0x0C0 (3/4 full).

New recommended value would be at 0x003.

FIFO_RX_PANIC_SHUTOFF, at offset 0x05c, is by default set at 0x080 (1/2 full).



New recommended value would be at 0x001.

Fix plan:

No plans to fix

FEC 3: FEC Management Data Clock is faster than IEEE maximum

Description: The FEC Management Data Clock (EC5_MDC) is much faster than described in the Reference Manual.

The intended frequency of the clock, as configured by the MIIMCFG[Mgmt Clock Select] field, is as follows:

- 000 - 1/4 of the eTSEC system clock divided by 8
- 001 - 1/4 of the eTSEC system clock divided by 8
- 010 - 1/6 of the eTSEC system clock divided by 8
- 011 - 1/8 of the eTSEC system clock divided by 8
- 100 - 1/10 of the eTSEC system clock divided by 8
- 101 - 1/14 of the eTSEC system clock divided by 8
- 110 - 1/20 of the eTSEC system clock divided by 8
- 111 - 1/28 of the eTSEC system clock divided by 8

Note: The eTSEC system clock is CCB Clock/2.

Based on this erratum, the actual frequency of the clock, as configured by the MIIMCFG[Mgmt Clock Select] field, is as follows:

- 000 - CCB clock divided by 6
- 001 - CCB clock divided by 6
- 010 - CCB clock divided by 10
- 011 - CCB clock divided by 14
- 100 - CCB clock divided by 17
- 101 - CCB clock divided by 24
- 110 - CCB clock divided by 34
- 111 - CCB clock divided by 49

Based on the above divider ratios, the minimum frequencies which can be obtained for EC5_MDC are:

- 12.24MHz (600MHz CCB)
- 10.88MHz (533MHz CCB)
- 8.16MHz (400MHz CCB)

Impact: Since the minimum EC5_MDC frequency is much faster than the IEEE maximum frequency of 2.5 MHz, the device may not function with certain PHYs.

Workaround: None

Fix plan: Fixed in Rev 2.0

GEN 1: External master cannot reset core

Description:	The Processor Core Initialization Register (PIR) bits P0 and P1 are intended to allow an external master to reset an individual core while maintaining the state of the platform and the other core. However, this feature does not work on certain revisions of silicon.
Impact:	On 8572 Rev 1.1.1 and earlier, an external master cannot reset a core while maintaining the state of the platform and the other core.
Workaround:	None
Fix plan:	<p>Fixed in 8572 Rev 2.0, with one additional requirement:</p> <p>Before setting PIR[P0], set the following bits: POWMGTCSR[Core0_IRQ_MSK] and POWMGTCSR[Core0_CI_MSK]</p> <p>Before setting PIR[P1], set the following bits: POWMGTCSR[Core1_IRQ_MSK] and POWMGTCSR[Core1_CI_MSK]</p> <p>Once the core has reset, clear these bits to allow re-enabling these interrupts.</p>

GPIO 1: Invalid data read from GPDAT bits corresponding to pins configured as outputs

Description:	GPIO pins can be configured as either inputs or outputs using the appropriate bits in the GPDIR register. Data read from GPDAT normally reflects the status of the corresponding GPIO pins. However, based on this erratum, reads to GPDAT will return invalid data for bits corresponding to pins configured as outputs. Note that although the correct value is driven on the pin, it is not verifiable by a read to GPDAT. For pins configured as inputs, reads to GPDAT will return valid data.
Impact:	Reads to GPDAT return invalid data for bits corresponding to pins configured as outputs. Therefore, it is not possible to use software to read-modify-write the GPDAT register.
Workaround:	Maintain a separate memory location which contains the value written to the GPIO outputs. When reading the status of the GPIO pins, read the status of the input pins from the GPDAT register, and read the status of the output pins from the separate memory location. Perform a bitwise OR between the two values to get the full status of the GPIO pins. Modify the value of the output pins as needed and write the result to both the GPDAT register and the separate memory location. For these read-modify-write sequences, another master must not modify GPDAT nor the separate memory location between the read and write operations.
Fix plan:	Fixed in Rev 2.0

I2C 1: 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 in Rev 1.1.1.

I2C 2: Enabling I²C could cause I²C bus freeze when other I²C devices communicate

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

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

Workaround: Use one of the following workarounds:

- Enable the I²C controller before starting any I²C communications on the bus. This is the preferred solution.
- If the I²C controller is configured as a slave, implement the following steps:
 - a. Software enables the device by setting I2CnCR[MEN] = 1 and starts a timer.
 - b. Delay for 4 I²C bus clocks.
 - c. Check Bus Busy bit (I2CnSR[MBB])

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

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

Fix plan: No plans to fix

JTAG 1: Boundary scan test on SerDes transmitter pins needs special requirement incompliant to IEEE 1149.1 specification

Description:	The IEEE 1149.1 EXTEST and CLAMP commands drive values stored in the programmable boundary scan cells onto the respective outputs and bidirectional pins. If these commands are run on the MPC8572E, the values driven out on the SerDes transmitter pins may be incorrect if any of the non-SerDes outputs and bidirectional pins are programmed to drive a logic 1.
Impact:	When running the IEEE 1149.1 EXTEST and CLAMP commands, the values read on the SerDes transmitter pins may not be the values expected.
Workaround:	<p>When running these commands, they should be performed in two passes if it is desired that all outputs and bidirectional pins are verified.</p> <p>One pass would be performed as originally intended reading all the non-SerDes pins and ignoring the SerDes transmitter outputs. In this pass, there is no requirement in regards to the value to be programmed on the boundary scan cell associated with any pin.</p> <p>Another pass would be performed reading all the SerDes transmitter pins and ignoring the non-SerDes outputs and bidirectional pins. In this pass, the boundary cells associated with all non-SerDes outputs and bidirectional pins should be programmed to drive a logic 0, resulting in the values programmed for the SerDes transmitter pins to be driven as intended.</p>
Fix plan:	Fixed in Rev 1.1.1

PCI-Ex 1: 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[19:24])(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 1.1.1

PCI-Ex 2: PCI Express LTSSM may fail to properly train with a link partner following 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: **Option 1** (for applications not booting from PCI Express)

If the link partner is not recognized, the PCI Express controller may be reset by performing the following procedure for both root complex and endpoint applications:

1. Set bit 4 of the reserved 32-bit register at CCSRBAR offset 0x0_AF00 (for PEX controller 1) or 0x0_9F00 (for PEX controller 2) or 0x0_8F00 (for PEX controller 3) while preserving the values of the other bits in the register. This can be done by ORing the contents of the reserved register with 0x0800_0000.
2. Wait 1 ms
3. Clear bit 4 of the reserved 32-bit register at CCSRBAR offset 0x0_AF00 (for PEX controller 1) or 0x0_9F00 (for PEX controller 2) or 0x0_8F00 (for PEX controller 3) while preserving the values of the other bits in the register. This can be done by ANDing the contents of the register with 0xF7FF_FFFF.

Option 2 (for applications booting from PCI Express)

This sequence resets the PCI Express controller only. In order to boot from PCI Express, the PCI Express controller must be reset during boot up using the boot sequencer, which is selected through the `cfg_boot_seq[0:1]` reset configuration signals. The boot sequencer should load configuration data to set and clear the above specified bit to reset the PCI Express controller before the host tries to configure the device. The following Configuration Preload Commands for the EEPROM data can be used.

```
7C 2B C0 88 40 00 00 //Sets the bit for the PCI Express
                        controller 1
7C 27 C0 88 40 00 00 //Sets the bit for the PCI Express
                        controller 2
7C 23 C0 88 40 00 00 //Sets the bit for the PCI Express
                        controller 3
7C 20 00 00 00 00 00 //Writes the reset value of PEX_CONFIG_ADDR
                        register to itself
7C 20 00 00 00 00 00 //Same as above to create 1 msec delay
7C 2B C0 80 40 00 00 //Clears the bit for the PCI Express
                        controller 1
7C 27 C0 80 40 00 00 //Clears the bit for the PCI Express
                        controller 2
```

7C 23 C0 80 40 00 00 //Clears the bit for the PCI Express
controller 3

Fix plan: Fixed in Rev 1.1.1

PCI-Ex 3: 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 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 HRESET is required.

Fix plan: Enhanced in Rev 1.1.1

As with previous revisions of silicon, the first transaction to the PCI-Express interface while the link is down will cause a Link Down Detected interrupt. It is advisable that software disable access to that PCI-Express link until the PEX_LTSSM_STAT returns a value of 0x16.

In silicon Rev 1.1 and later, every Load to a PCI-Express link which is down will cause the associated instruction to stall, meaning that interrupts can still be processed while this instruction is pending. Setting HID1[RXFE] allows a Machine Check to be generated for each Load that is performed to the PCI-Express interface while the link is down. Each time a Machine Check is detected, software should check the Machine Check Syndrome Register (MCSR) and the Machine Check Address Register (MCAR). If a Load to a PCI-Express link which is down has occurred, the MCSR will indicate BUS_RBERR, and the address in the MCAR will fall within the PCI-Express region. The Machine Check interrupt handler should then prevent further accesses to the PCI-Express link until it has successfully trained again, as indicated by a value of 0x16 in the PEX_LTSSM_STAT register.

Note that this procedure requires that MSR[ME] and HID1[RXFE] are set.

PCI-Ex 4: 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.

>Rev 1.0 only:

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.

>Rev 1.1.1 only:

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.

>Rev 1.0 and 1.1.1:

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:

Rev 1.0 only:

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.

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 3, 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.

Rev 1.1.1 only:

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

Rev 1.0 and 1.1.1:

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

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.

Rev 1.1.1 only:

Once the link is confirmed down, register accesses can be enabled by writing 0x80C0_0000 to the PCI Express memory-mapped debug mode register at offset 0xF00 followed by polling the PEX_IP_BLK_REV1 register until it returns non-Fs to confirm that the reset of pending transactions is complete. Once software has completed any needed register accesses, it should return to normal function by writing 0x8040_0000 to the PCI Express debug register at offset 0xF00.

Be aware that while the PCI Express controller is in the debug mode that enables normal register accesses, new reads or writes to PCI Express are not canceled. Software must ensure that no new accesses are sent to PCI Express while this debug mode is enabled.

Fix plan:

No plans to fix

PCI-Ex 5: 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:
$$500 \text{ MHz} \times (\text{PCI-Express link width})/8.$$

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:
$$527 \text{ MHz} \times (\text{PCI-Express link width})/8.$$

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 occurring 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.

Fix plan: The hardware specification documents reflect these platform frequency requirements.

PME 1: DXCM Parity Malfunction may cease operation of PME

Description:	<p>The Pattern Matcher Engine (PME) logic consists of a Data Examination Engine (DXE) with an 80-entry deep Data Examination Context Memory (DXCM). The DXCM is used as a backtracking stack with 96 bits of data as well as 4 bits of parity. The whole DXCM stack is indexed by a 7-bit stack pointer logic.</p> <p>The implementation includes a prefetch circuit that reads the top entry of the stack into a holding register. As entries are removed from the stack, the new “top” entry is pre-fetched into this holding register.</p> <p>The problem occurs when the DXE logic retrieves the last entry from the stack. The last entry is taken from the holding register and the prefetch circuit tries to retrieve the next entry, resulting in a wraparound and an out-of-range access that returns arbitrary data that is not used but may cause a parity error to occur.</p>
Impact:	<p>When this problem occurs, the PME stops scanning the current work unit, gracefully shut down the channel, and report the error to software. Based on this, software is able to determine which work units have been scanned successfully, which work units need to be re-scanned, and which work units have yet to be scanned.</p> <p>Re-scanning the same work unit, however, against the same set of patterns cause the same parity error.</p>
Workaround:	<p>The stack is only used for expressions with an alternate link, a “cautious” repeat, or a “lazy” repeat. Expressions such as <code>/Za*?b/</code> or <code>/abc(d{2} e{3})/</code> cause the problem to occur.</p> <p>A work around has been implemented in the compiler to identify most of these “problematic” expressions and to automatically insert instructions when compiling these expressions to cause the DXE to place a dummy entry as the first entry in the stack. This prevents the prefetch circuitry from accessing the wraparound location, because the dummy entry would be the last entry that would be pre-fetched.</p> <p>This work around has no side effects other than using up instruction space that could otherwise have been used for the expression. The compiler can use up to 30 “testlines” to represent an expression. This work around consumes two of those “testlines,” resulting in a total of 28 usable “testlines.” Very long expressions with an alternate link, a “cautious” repeat, or a “lazy” repeat, requiring more than 28 “testlines” would need to be split into two expressions.</p> <p>The work around handles most practical situations where this problem can occur. There are, however, theoretical situations that are not covered by this work around, which may result in the need for certain expressions to be manually rewritten. These are identified by an explicit compile error during compile time.</p>
Fix plan:	<p>Fixed in Rev 1.1.1.</p>

SEC 1: Security registers from offset 0x3_1510 to 0x3_1518 cannot be accessed with 4-byte accesses

Description:	Security registers xyz are described in the reference manual as registers that can be accessed with 4- or 8-byte accesses. The option to access these registers with 4-byte access is broken.
Impact:	These new performance counters cannot be accessed with 4-byte accesses.
Workaround:	Use 8-byte access when accessing registers 0x3_1510 to 0x3_1518.
Fix plan:	No plans to fix

SEC 2: Protocol descriptor hang conditions

Description:	The SEC is designed to perform single pass encryption and integrity checking as required for security protocols including IPsec and SSL/TLS. When using the descriptor types required for single pass IPSEC and SSL/TLS, where a primary EU is selected, and no secondary EU is selected, the channel hangs rather than producing an illegal descriptor header error.
Impact:	<p>Channel hangs if no secondary EU is selected for single pass IPSEC and SSL/TLS. There are two specifically observed hang conditions, as follows:</p> <ol style="list-style-type: none">1. IPsec inbound without a secondary EU (Outbound, with or without a secondary EU is fine; inbound with a secondary EU is fine). The channel behavior is not predictable and may hang.2. TLS_SSL_STREAM inbound if the primary EU is AFEU and there is No secondary EU, the channel hangs.
Workaround:	Single pass protocol descriptors are designed for simultaneous encryption and integrity checking, and operation which required two EUs. For IPsec or SSL with integrity only, other descriptor types should be used. Note that this errata does not impact IPsec with AES-GCM (a use of IPsec with a single primary EU). IPsec with AES-GCM uses a different descriptor type.
Fix plan:	No plans to fix

SEC 3: AES-GCM IV Length Restriction

Description:	<p>Like most other modes of AES, AES-GCM uses a key and an IV to transform plaintext to ciphertext. The length of the IV is standardized by multiple independent bodies. The base AES-GCM specification allows the use of any size of IV or AAD or plaintext/ciphertext.</p> <p>Other standards bodies, including the IEEE and IETF, define security protocols which use AES-GCM, such as MACSec (IEEE Std 802.1ae™-2005), and IPSec. The specifications for the use of AES-GCM within these security protocols restricts the size of the IV, AAD, and plaintext/ciphertext.</p> <p>The AESU in the SEC performs AES-GCM without any issue whenever the IV is 96 bits, however the AESU fails under the following conditions:</p> <ol style="list-style-type: none">1. The IV length is not 96 bits.2. The initial IV Ghash operation produces a value with the 32 least significant bits all 1's
Impact:	<p>Both IPSec and MACSec (IEEE 802.1ae) specify an IV length of 96 bits, which allows the SEC to support these protocols without restriction. The only known concern associated with this errata comes from the current draft standard for P1619.1 (Draft Standard for Authenticated Encryption with Length Expansion for Storage Devices), which permits IVs other than 96 bits in length. Because there is no way to predict whether the initial IV Ghash operation produces a value with the 32 least significant bits all 1's, the SEC should not be used for AES-GCM in P1619.1, or any other protocol which does not limit IV length at 96 bits.</p>
Workaround:	<p>For protocols specifying 96b IVs, no work around is necessary. For protocols specifying other IV lengths, none is possible.</p>
Fix plan:	<p>No plans to fix</p>

SEC 4: TLS_SSL_Block_Inbound HMAC Error

Description:

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

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

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

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

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

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

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

Impact:

TLS_SSL_Block_Inbound operations produce incorrect HMAC values.

Workaround:

- Option 1: For TLS_SSL_Outbound operations, use the SEC's single pass descriptor type 1000_1 for maximum performance. For TLS_SSL_Inbound, use two descriptors:
 - First descriptor: Use type 0001_0 'common non-snoop', with header set for the appropriate block cipher algorithm, to decrypt the payload through Pad Length. Upon completion of this descriptor, software changes the value of the Length field in the record header and launches the second descriptor.
 - Second Descriptor: Type 0001_0 'common non-snoop' set for the appropriate HMAC algorithm (for example, HMAC-SHA-1). The

pointers in this descriptor tell the SEC to calculate an HMAC over the SSL Record Header||Payload. The SEC can be set to automatically compare the generated HMAC against the received HMAC, or the descriptor can write the generated HMAC to memory for a comparison by software.

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

Fix plan: No plans to fix

SEC 5: Non-compliant implementation of deterministic pseudo-random number generator

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

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

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

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

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

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

Fix plan: No plans to fix

SEC 6: Limitations on Custom Modes of CRC

Description:	<p>The CRCU supports 2 standard CRCs, as well as custom CRCs in which the user defines the polynomial and bit manipulations associated with the polynomial. These bit manipulations are bit swapping, byte swapping, and complementing the output of the polynomial. Using the RAW bit in the CRCU Mode Register, the user controls whether all or none of the manipulations are performed.</p> <p>The CRCU supports generation of correct CRC output for any polynomial that uses either the raw polynomial output, or an output that is bit swapped, byte swapped, and complemented. It does not support CRCs in which some bit manipulations are performed, but not others.</p>
Impact:	<p>The OFDMA CRC used in Wimax is not supported, due to the fact that it requires some, but not all, bit manipulations.</p>
Workaround:	<p>Perform the OFDMA CRC in software.</p>
Fix plan:	<p>No plans to fix</p>

SEC 7: 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

SRIO 1: Accessing SRIO memory mapped space causes the device to hang when SRIO is disabled via POR configuration

Description:	When SRIO is disabled via the POR configuration (cfg_IO_ports[0:3]), accessing SRIO memory map space causes the device to hang.
Impact:	The device hangs if SRIO memory mapped space is accessed when SRIO is disabled via POR configuration (cfg_IO_ports[0:3]).
Workaround:	<p>Option 1:</p> <p>Do not access SRIO memory map if it is disabled via the POR configuration (cfg_IO_ports[0:3]).</p> <p>Option 2:</p> <p>Enable the SRIO via the POR configuration (cfg_IO_ports[0:3]). During software boot sequence, disable the SRIO via the Device Disable Register. This prevents device hangs when accessing the disabled SRIO block, assume the valid SerDes 1 reference clock is provided.</p>
Fix plan:	No plans to fix

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

Description:	<p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>
Impact:	<p>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.</p>
Workaround:	<p>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.</p> <p>To disable error rate counting for unexpected AckIDs clear PnERECSR[UA] (Port n Error Rate Enable Command and Status Register - Unexpected AckID).</p>
Fix plan:	<p>No plans to fix</p>

SRIO 3: Message unit cannot generate messages with priority 0

Description:	<p>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:</p> <p>OMnDATR[DTFLOWLVL]</p> <p>00 SRIO transaction priority of 1</p> <p>01 SRIO transaction priority of 1</p> <p>10 SRIO transaction priority of 2</p> <p>11 Reserved</p> <p>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.</p>
Impact:	<p>Outbound messages have a higher priority than expected if the programmed flow level is 00.</p>
Workaround:	<p>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.</p>
Fix plan:	<p>No plans to fix</p>

UART 1: BREAK detection triggered multiple times for a single break assertion

Description:	<p>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.</p> <p>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.</p> <p>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.</p>
Impact:	<p>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.</p>
Workaround:	<p>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:</p> <ol style="list-style-type: none">1. Read URBR, which returns a value of zero, and clears the ULSR[DR] bit2. Delay at least 1 character period3. Read URBR again, which return a value of zero, and clears the ULSR[DR] bit <p>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.</p> <p>This workaround requires that the break signal be at least 2 character-lengths in duration.</p> <p>This work around applies to both polling and interrupt-driven implementations.</p>
Fix plan:	<p>No plans to fix</p>

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

<http://www.freescale.com/support>

USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor
Literature Distribution Center
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

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

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale and the Freescale logo are trademarks or registered trademarks of Freescale Semiconductor, Inc. in the U.S. and other countries. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© Freescale Semiconductor, Inc., 2009-2010. All rights reserved.

Document Number: MPC8572ECE

Rev. K
2/2010

