MPC8349E Chip Errata

This document details all known silicon errata for the MPC8349E family of chip errata. The following table provides a revision history for this document.

Table 1. Document Revision History

Revision	Date	Significant Changes
19	08/2011	• Added CPU-A022, SEC-A001, USB-A001, USB-A002, USB-A003, USB-A005, USB-A007 • Modified General12, General17 workaround, PCI15, PCI20 workaround, USB32 workaround, SEC7 impact • Updated Fix plan for DDR8, DDR9, DDR10, PCI12, LBC3, SEC4, SEC5. Had been "Functionality will be added to silicon revision 3.x" now "Functionality added to silicon revision 3.x." • Updated Fix plan for General4. Had been "Scenario 1 will be fixed in revision 1.1 Scenario 2 will be fixed in revision 3.x" now "Scenario 1 fixed in Rev 1.1 Scenario 2 fixed in Rev 3.x" • Updated Fix plan for USB21, USB22, JTAG3. Had been "Under Review" now "No plans to fix."
18	7/2009	• Added USB31–USB38. • Added a NOTE to USB24.
17	2/2009	• Added USB24, USB25, I2C1, and General17. • Removed “This is only a potential errata...” from EDAauxxxxx LBC2 project solution.
16	2/2008	• Added USB21, USB22, General12 • Modified JTAG3 workaround. • Added LBC1 • Added Rev 3.1 mask number.
15	3/2007	• Fixed cross reference for CPU5 in “Summary of MPC8349E Silicon Errata and Applicable Revision” table. • Edited all projected fix plans. • Added PCI20
14	2/2007	• Added PCI19
13	2/2007	• Added JTAG3.

Table continues on the next page...

Table 1. Document Revision History (continued)

Revision	Date	Significant Changes
12	2/2007	• Added USB17, TSEC6.1 and PCI15. • Fixed CPU4 and CPU4 links in summary table. • Changed all references of rev 3.0 & 3.1 to 3.x. • Corrected General5 in summary table to apply to rev 3.x silicon.
11	10/2006	• Added DDR14, CPU5, and DMA2 • Added PCI14 • Added information about device ID related to silicon version number on page 1 and in Table 2, “MPC8349E Family Devices and Silicon Revisions”
10	8/2006	• Added USB15, RESET3, and General6
9	5/2006	• Added SEC6 and SEC7 • Updated GPIO1. • Added LBC5 • Updated TSEC22 • Modified code sample in LBC2 to correct register name. LBDLLCR was referenced; correct name of register is DLLOVR • Added PCI13 and DDR12. Modified TSEC7
8	2/2006	• Addition of TSEC21, TSEC22 and DDR6
7	12/2005	• Modified SPI1 error to reflect FIFO support of 2 characters and not 3 • Removed errata items relating to LPDDR (were numbered DDR6 and DDR7) • Addition of ARB3, PCI11, SPI4, and TSEC20
5	9/2005	• Addition of G5
4	9/2005	• Addition of USB14 and SEC3
3	8/2005	• Addition of TSEC7 and PCI11
2	6/2005	• Addition of CPU3, CPU4, and RESET2
1	5/2005	• Addition of PCI10
0	5/2005	• Initial release

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

Table 2. MPC8360E Family Devices and Silicon Revisions

Device	Silicon Revision	1.0	1.1	3.1
	Mask	MPC834xE	MPC834xE	MPC834xE
		0K52M	1K52M	2M12E
MPC8349E		√	√	√
MPC8347E		√	√	√
MPC8343E		√	√	√

Table 3 summarizes all known errata and lists the corresponding silicon revision level to which they apply. A 'Yes' entry indicates the erratum applies to a particular revision level, and an 'No' entry means it does not apply.

Table 3. Summary of Silicon Errata and Applicable Revision

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	3.x
CPU					
CPU1	CPU hangs when a load from a cache inhibited space is followed by a Trap Word Immediate instruction (twi, tw) under certain conditions (abort on speculative load)	Fixed in Rev 3.x	Yes	Yes	No
CPU2	Incorrect **dcbi** and **dcbf** operation during load and store operation	Fixed in Rev 3.x	Yes	Yes	No
CPU3	A misaligned access across a page boundary into a protected page may result in incorrect values for DAR	Fixed in Rev 3.x	Yes	Yes	No
CPU4	Access to illegal address might cause core to hang	Fixed in Rev 3.x	Yes	Yes	No
CPU5	Cache instruction **dcbz** canNOT allocate any tag in data cache when data cache is locked.	Fixed in Rev 3.x	Yes	Yes	No
CPU-A022	The e300 core may hang while using critical interrupt	No plans to fix	Yes	Yes	Yes
DDR					
DDR3	DDR memory controller does not guarantee that the queue is empty before going to self refresh	Fixed in Rev 3.x	Yes	Yes	No
DDR4	DDR memory controller violates the JEDEC spec when exiting self-refresh state	Fixed in Rev 1.1	Yes	No	No
DDR5	64-bit DDR with enabled auto-precharge mode will violate t_{rp} if optimal timing parameters are used	Fixed in Rev 3.x	Yes	Yes	No
DDR6	The 834x DDR controller may not receive correct data during Read cycles from DDR SDRAM.	Fixed in Rev 3.x	Yes	Yes	No
DDR7	834x DDR controller does not meet timings for MDQS-MDQ/MECC inputs skew per byte as stated in MPC834x hardware specification	Fixed in Rev 3.x	Yes	Yes	No
DDR8	Signal SELF_REFRESH_PANIC_REQUEST not available	Functionality added to silicon revision 3.x	NE	NE	E
DDR9	DDR2 support not available	Functionality added to silicon revision 3.x	NE	NE	E
DDR10	Signals MODT[0:3], MBA2, MDIC[0:1] are not available	Functionality added to silicon revision 3.x	NE	NE	E
DDR12	DDR input crossing-point voltage does not meet JEDEC specifications	Fixed in Rev 3.x	Yes	Yes	No
DDR14	The automatic CAS-to-Preamble feature of the DDR controller can calibrate to incorrect values	No plans to fix	No	No	Yes

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	3.x
I2C					
I2C1	Enabling I^2C could cause I^2C bus freeze when other I^2C devices communicate	No plans to fix	Yes	Yes	Yes
General					
General2	Write access to a module with clock disabled causes the device to hang	Fixed in Rev 3.x	Yes	Yes	No
General3	CKSTP/LDP pin function reset value	Fixed in Rev 1.1	Yes	No	No
General4	I^2C boot sequencer may fail if $\overline{\text{HRESET}}$ happens when SDA is driven low	There are two scenarios	Yes	Partial	No
General5	Additional 256K clock cycles of $\overline{\text{HRESET}}$ signal	No plans to fix	No	No	Yes
General6	Override of HRCW word via JTAG when source is the I2C bus may not work if EEPROM on I2C bus contains an incorrect preamble	Fixed in Rev 3.x	Yes	Yes	No
General12	CSB deadlock	No plans to fix	Yes	Yes	Yes
General17	DUART: Break detection triggered multiple times for a single break assertion	No plans to fix	Yes	Yes	Yes
TSEC					
TSEC5	TSEC in RGMII/RTBI mode does not receive data correctly	Fixed in Rev 1.1.	Yes	No	No
TSEC6	TSEC receiver FIFO overflow can cause the receiver to drop frames or hang (Case 1)	Fixed in Rev 1.1.	Yes	No	No
TSEC6.1	TSEC receiver FIFO overflow can cause the receiver to drop frames or hang (Case 2)	No plans to fix	Yes	Yes	Yes
TSEC7	RGMII timing does not meet with RGMII specification	No plans to fix	Yes	Yes	Yes
TSEC20	TBI transmit AC timing specification tTTKHDX is not met	Fixed in Rev 3.x	Yes	Yes	No
TSEC21	The MPC834x does not meet the TSEC GMII input hold timing	Fixed in Rev 3.x	Yes	Yes	No
TSEC22	The power supply of TSEC1, LVDD1, is internally tied to the power supply of TSEC2, LVDD2	Fixed in Rev 3.x	Yes	Yes	No
PCI					
PCI2	PCI returns bad data on a master read following perr_response assertion	Fixed in Rev 3.x	Yes	Yes	No
PCI3	Data corruption when bus arbiter is parked to PCI and external PCI master performs transactions with non-consecutive byte enables	Fixed in Rev 3.x	Yes	Yes	No
PCI4	Data corruption if PCI is accessed as CSB slave device and CPU snoops	Fixed in Rev 3.x	Yes	Yes	No
PCI5	Target read transactions of more than one cache line from prefetchable memory space may cause data corruption	Fixed in Rev 1.1	Yes	No	No
PCI6	PCI host configuration read from unused slot terminates with CSB error	Fixed in Rev 1.1	Yes	No	No

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	3.x
PCI7	PCI subsystem ID registers are writable from the PCI side	Fixed in Rev 1.1	Yes	No	No
PCI8	CFG_CLKIN_DIV cannot be set high in PCI agent mode	Fixed in Rev 1.1	Yes	No	No
PCI9	The device can hang if PCI inbound transaction or DMA transaction with snoop enabled happens at the same time as the CPU tries to access PCI outbound window	Fixed in Rev 3.x	Yes	Yes	No
PCI10	PCI inbound write data may be corrupted if a CSB/PCI clock ratio of 2:1 is used	Fixed in Rev 3.x	Yes	Yes	No
PCI11	Sub class ID setting is wrong in the host mode	Fixed in Rev 3.x	Yes	Yes	No
PCI12	PCI Rev 2.3 is not available	Functionality added to silicon revision 3.x	NE	NE	E
PCI13	Data corruption occurs when an external PCI initiator issues the “Read Multiple” command	Fixed in Rev 3.x	Yes	Yes	No
PCI14	PCI input setup time will be changed to meet PCI AC timing specification	No plans to fix	Yes	Yes	Yes
PCI15	Assertion of $\overline{\text{STOP}}$ by a target device on the last beat of a PCI memory write transaction can cause a hang	No plans to fix	Yes	Yes	Yes
PCI19	Dual-address cycle inbound write accesses can cause data corruption	No plans to fix	Yes	Yes	Yes
PCI20	PCI controller may hang when returning from "PCI pins low" state	No plans to fix	Yes	Yes	Yes
LBC					
LBC1	UPM does not have indication of completion of a RUN PATTERN special operation	No plans to fix	Yes	Yes	Yes
LBC2	LBC clock glitch when performing synchronous accesses with DLL enabled	No plans to fix	Yes	Yes	No
LBC3	Chip select pins 4–7 not available (LCS[4:7])	Functionality added to silicon revision 3.x	NE	NE	E
LBC5	$\overline{\text{LCS}}$[6:7]/PCI_CLK[3:4] drive zeros during reset	No plans to fix	No	No	Yes
SPI					
SPI1	Third character received causes overflow if the core does not read first and second characters	Fixed in Rev 3.x	Yes	Yes	No
SPI2	SPIE[NF] is set after transmission of a one byte block	Fixed in Rev 3.x	Yes	Yes	No
SPI3	SPIE[NF] bit not set after writing the first character	Fixed in Rev 3.x	Yes	Yes	No
SPI4	LT bit in SPI event register is never set	Fixed in Rev 3.x	Yes	Yes	No
USB					
USB1	USB data corruption after read burst in which system bus error was generated	Fixed in Rev 3.x	Yes	Yes	No

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	3.x
USB3	MPH USB port 1 will not operate without MPH0_CLK signal present on port 0	Fixed in Rev 1.1	Yes	No	No
USB4	ULPI interface does not recognize rx_active immediate	Fixed in Rev 1.1.	Yes	No	No
USB5	RX EOP Detection Error	Fixed in Rev 1.1.	Yes	No	No
USB6	Transmit to transmit inter-packet gap	Fixed in Rev 1.1.	Yes	No	No
USB7	ULPI Register read commands corrupt the receive packet	Fixed in Rev 3.x	Yes	Yes	No
USB8	Early LS Handshake Time-out	Fixed in Rev 3.x	Yes	Yes	No
USB9	FS EOP insertion on pre-PID	Fixed in Rev 3.x	Yes	Yes	No
USB10	Test J/Test K mode configuration	Fixed in Rev 3.x	Yes	Yes	No
USB11	Low power resume wait delay	Fixed in Rev 3.x	Yes	Yes	No
USB12	Stall bit setting during setup lockout	Fixed in Rev 3.x	Yes	Yes	No
USB13	Host lock up on the disconnect	Fixed in Rev 3.x	Yes	Yes	No
USB14	Port numbering in Queue Head violates EHCI specification	Fixed in Rev 3.x	Yes	Yes	No
USB15	Read of PERIODICLISTBASE after successive writes may return a wrong value in host mode	No plans to fix	No	No	Yes
USB17	USBDR and MPH do not generate interrupt upon CRC16 error in IN data packet when in host mode.	No plans to fix	Yes	Yes	Yes
USB21	SE0_NAK issue	No plans to fix	Yes	Yes	Yes
USB22	USB clock to output valid allows less margin on input setup time	No plans to fix	Yes	Yes	Yes
USB24	A write to some USB registers such as PORTSC can cause the system to hang when no USB PHY clock is applied	No plans to fix	Yes	Yes	Yes
USB25	In host mode, when the software forces a port resume by writing into the FPR bit of the portsc register, the port change detect interrupt bit is falsely fired	No plans to fix	Yes	Yes	Yes
USB31	Transmit data loss based on bus latency	No plans to fix	Yes	Yes	Yes
USB32	Missing SOFs and false babble error due to Rx FIFO overflow	No plans to fix	Yes	Yes	Yes
USB33	No error interrupt and no status will be generated due to ISO mult3 fulfillment error	No plans to fix	Yes	Yes	Yes
USB34	NAK counter decremented after receiving a NYET from device	No plans to fix	Yes	Yes	Yes
USB35	Core device fails when it receives two OUT transactions in a short time	No plans to fix	Yes	Yes	Yes
USB36	CRC not inverted when host under-runs on OUT transactions	No plans to fix	Yes	Yes	Yes
USB37	OTG Controller as Host does not support Data-line Pulsing Session Request Protocol	No plans to fix	Yes	Yes	Yes

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	3.x
USB38	USB Controller locks after Test mode "Test_K" is completed	No plans to fix	Yes	Yes	Yes
USB-A001	Last read of the current dTD done after USB interrupt	No plans to fix	Yes	Yes	Yes
USB-A002	Device does not respond to INs after receiving corrupted handshake from previous IN transaction	No plans to fix	Yes	Yes	Yes
USB-A003	Illegal NOPID TX CMD issued by USB controller with ULPI interface	No plans to fix	Yes	Yes	Yes
USB-A005	ULPI Viewport not Working for Read or Write Commands With Extended Address	No plans to fix	Yes	Yes	Yes
USB-A007	Host controller fails to enter the PING state on timeout during High Speed Bulk OUT/DATA transaction	No plans to fix	Yes	Yes	Yes
SEC					
SEC1	Global soft reset does not work when interrupts are pending	Fixed in Rev 3.x	Yes	Yes	No
SEC2	Public key execution unit does not work	Fixed in Rev 1.1	Yes	No	No
SEC3	AESU Input FIFO error on less than 16B data size	Fixed in Rev 3.x	Yes	Yes	No
SEC4	XOR parity generation accelerator for RAID applications	Functionality added to silicon revision 3.x	NE	NE	E
SEC5	SHA-224 support is unavailable	Functionality added to silicon revision 3.x	NE	NE	E
SEC6	AES-CTR mode data size error	No plans to fix	Yes	Yes	Yes
SEC7	Single descriptor SRTP error	No plans to fix	Yes	Yes	Yes
SEC-A001	Channel Hang with Zero Length Data	No plans to fix	Yes	Yes	Yes
JTAG					
JTAG1	Under JTAG mode, the pin type of PCI1_INTA and MCP_OUT is wrong	Fixed in Rev 3.x	Yes	Partial	No
JTAG2	Under JTAG mode, pin type of certain pins is wrong	Fixed in Rev 1.1	Yes	No	No
JTAG3	TRST VIH input level may glitch and cause TAP controller to reset.	No plans to fix	Yes	Yes	Yes
ARB					
ARB1	Bus hangs under certain transaction pattern	Fixed in Rev 1.1	Yes	No	No
ARB3	Changing the value of ACR[PIPE_DEP] can cause the arbiter to generate false data time out	No plans to fix	Yes	Yes	Yes
IPIC					
IPIC1	Feature that the highest priority interrupt and two highest priority interrupts from each group can be programmed to support $\overline{cint}$ and $\overline{smi}$ does not work	Fixed in Rev 3.x	Yes	Yes	No

Table continues on the next page...

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

Errata	Name	Projected Solution	Silicon Rev.		
			1.0	1.1	3.x
DMA					
DMA1	DMA writes to prefetchable buffer in IOS might cause memory coherence problem	Fixed in Rev 1.1	Yes	No	No
DMA2	Data corruption by DMA when destination address hold (DAHE) bit is used	No plans to fix	Yes	Yes	Yes
RESET					
RESET1	$\overline{\text{HRESET}}$ is not synchronized prior to use in the internal state machines	Fixed in Rev 1.1	Yes	No	No
RESET2	Change of LBIUCM or DDRCM bit value in reset configuration word will cause the device not able to recover from $\overline{\text{HRESET}}$	Fixed in Rev 3.x	Yes	Yes	No
RESET3	External Soft reset functionality is not functional	No plans to fix	Yes	Yes	Yes
PMC					
PMC1	Entering low power mode may result in loss of data in DDR memory	Fixed in Rev 1.1	Yes	No	No
THERM					
THERM1	THERM0 pin does not work	Fixed in Rev 1.1	Yes	No	No
GPIO					
GPIO1	Certain GPIO pins are not functioning if TSEC1 is using RGMII and RTBI mode	Fixed in Rev 3.x	Yes	Yes	No

CPU1: CPU hangs when a load from a cache inhibited space is followed by a Trap Word Immediate instruction (twi, tw) under certain conditions (abort on speculative load)

Description: Devices: MPC8349E, MPC8347E, MPC8343E

CPU hangs while speculatively executing a load instruction following a system trap instruction (**twi**, **tw**) that triggers a system trap exception. The problem happens only under the following conditions:

- D-cache is on.
- Data Cache is inhibited (WIMG).
- Memory is not guarded.
- The trap instruction and load instruction are dispatched in the same clock cycle.

Impact: Core hangs under this particular scenario.

Workaround: Set the guarded bit for cache inhibited memory space.

Fix plan: Fixed in Rev 3.x.

CPU2: Incorrect dcbi and dcbf operation during load and store operation

Description: Devices: MPC8349E, MPC8347E, MPC8343E

A load or store occurring immediately after a **dcbi** or **dcbf** in the same 32-byte cache block may operate as a hit in the data cache. The correct operation is for that load or store to miss in the data cache and access data from memory instead.

The problem happens only under the following condition:

Software that uses **dcbi** and **dcbf** to manually manage the coherency of the data cache, rather than relying on the automatic hardware coherency of snoops.

Impact: Incorrect coherency if managed by software.

Workaround: Follow a **dcbi** or **dcbf** with a sync instruction if the following load or store could be to the same 32-byte cache block.

Fix plan: Fixed in Rev 3.x.

CPU3: A misaligned access across a page boundary into a protected page may result in incorrect values for DAR

Description: Devices: MPC8349E, MPC8347E, MPC8343E

A misaligned access across a page boundary into a protected page will result in incorrect values for DAR[30,31] (incorrect byte address).

Impact: DAR is normally used for debug to identify the misaligned access address. Users may get the wrong address from DAR. Routines of this type will rerun the instruction which causes the DSI misalignment exception resulting in correct operation.

Workaround:
1. Always set bits DAR[30,31] to zero during any DSI exception.
2. Always rerun instruction after a misaligned access.

Fix plan: Fixed in Rev 3.x.

CPU4: Access to illegal address might cause core to hang

Description: Devices: MPC8349E, MPC8347E, MPC8343E

A read access to unmapped illegal address results in the system generating a TEA assertion. The assertion of TEA will generate a machine check exception. Due to a logic error, the TEA may result in a core deadlock condition.

Impact: System might not recover from the TEA which is either caused by a faulty hardware system condition or software searching the memory map for valid/invalid memory locations. The system has to be HRESET.

Workaround: None

Fix plan: Fixed in Rev 3.x.

CPU5: Cache instruction dcbz canNOT allocate any tag in data cache when data cache is locked.

Description: Devices: MPC8349E, MPC8347E, MPC8343E

For a **dcbz** operation, the PLRU replacement algorithm will be used to determine which way of the cache should be used for a cache update of zero's on a cache miss, regardless of the HiD0[dlock] setting. The HiD2[d/i way-lock] bits are used to direct the dcbz allocation toward the higher numbered ways.

In MPC8349E rev 1.*x,* the **dcbz** instruction does not ignore the LOCK status of the D-Cache (HID0[DLOCK] = 1), and apparently nothing happens (no error or exception occurs).

Impact: The **dcbz** instruction does not work.

Workaround: No.

Fix plan: Fixed in Rev 3.x.

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

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

Sample Code:

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

Fix plan: No plans to fix

DDR3: DDR memory controller does not guarantee that the queue is empty before going to self refresh

Description: Devices: MPC8349E, MPC8347E, MPC8343E

DDR memory controller does not guarantee that the queue is empty before going to self-refresh.

When the device exits from the low power mode, it executes the transactions left in the queue. But in the case where the power supply is turned off, the transactions left in the queue are lost.

Impact: Write data may not reach memory.

Workaround: The following sequence empties the queue:

1. Read address A where A is a DDR memory address.
2. Write the same data to A; do not change the content of A.
3. Flush A with **dcbf** if data cache is enabled.
4. Read A.

The last read to ‘A’ causes an address collision with the previous write to ‘A’ which results in all writes ahead of ‘A’ to be drained to memory before the data for the last read to ‘A’ is returned from memory.

Fix plan: Fixed in Rev 3.x.

DDR4: DDR memory controller violates the JEDEC spec when exiting self-refresh state

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When the chip goes into low power mode and DDR self-refresh is enabled, under certain conditions the DDR controller can produce exiting self-refresh commands even when the MCK clock is off for a few cycles.

Impact: JEDEC standard is violated. The standard requires that the clock be stable prior to exiting from the self-refresh state.

Workaround: None

Fix plan: Fixed in Rev 1.1

DDR5: 64-bit DDR with enabled auto-precharge mode will violate t_{rp} if optimal timing parameters are used

Description: Devices: MPC8349E, MPC8347E, MPC8343E

For a given set of programmed timing parameters to the DDR memory controller, the DRAM's t_{rp} specification may be violated. The parameters involved are the following:

- ACTTOPRE TIMING_CFG_1[4:7]
- ACTTORW TIMING_CFG_1[9:11]

The violation of t_{rp} happens if ACTTOPRE > ACTTORW + 2

For instance, with 64-bit DDR 266 MHz, ACTTOPRE = 6, ACTTORW = 3, it satisfies the inequality, thus, the problem occurs.

The violation happens only when a PRECHARGE ALL command is issued by the DDR memory controller (to prepare for a refresh cycle) right after a read with the auto-precharge command.

Impact: May potentially corrupt the DRAM content.

Workaround: Disable auto-precharge mode by setting DDR_SDRAM_INTERVAL[BSTOPRE] to a non-zero value and CSn_CONFIG[AP_n_EN] to 0. Note that even though a larger ACTTORW can be used to work around the problem, this is not recommended as it has a significant performance impact on any DDR read/write access.

Fix plan: Fixed in Rev 3.x.

DDR6: The 834x DDR controller may not receive correct data during Read cycles from DDR SDRAM.

Description: Devices: MPC8349E, MPC8347E, MPC8343E

After issuing a Read cycle to DRAM, the 834x input receivers for the DDR I/Os will be enabled. However, depending upon board delays, CAS latency, and frequency, there may not be enough margins from when the Input Enables are asserted until data is presented at the 834x device.

Impact: Data could be corrupted when it is read from the DDR SDRAM.

Workaround: Input Enables must be turned on two clock cycles earlier. The two clock cycles penalty will only affect performance when switching between different chip selects. Performance will not be affected if only one chip select is used.

The following shows the value that must be written to the reserved register at offset IMMRBAR + 0x2f00 based on the CAS latency.

CAS Latency	Enable 2 Cycles Earlier
2 Cycles	0x201c_0000
2.5 Cycles	0x202c_0000
3 Cycles	0x202c_0000

NOTE

The reserved register stated above will not be supported in future silicon revisions.

Fix plan: Fixed in Rev 3.x.

DDR7: 834x DDR controller does not meet timings for MDQS-MDQ/MECC inputs skew per byte as stated in MPC834x hardware specification

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The DDR controller does not meet the t_{DISKEW} timing MDQS-MDQ/MECC inputs skew per byte. The table below lists the new t_{DISKEW} timing.

Frequency	Current T_{DISKEW} Specification	New T_{DISKEW} Specification
333Mhz	750ps	700ps
266Mhz	1125ps	750ps

Impact: Data will be corrupted if t_{DISKEW} is not met under worst case conditions.

Workaround: Follow the guidelines that are outlined in application note AN2582 *Hardware and Layout Design Considerations for DDR Memory Interfaces*. AN2582 discusses the hardware and layout design considerations for DDR memory interfaces. This application note should be used to as a guide to understand the impact.

Fix plan: Fixed in Rev 3.x.

DDR8: Signal SELF_REFRESH_PANIC_REQUEST not available

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Panic interrupt support is not available.

Impact: Cannot use panic interrupt.

Workaround: None

Fix plan: Functionality added to silicon revision 3.x

DDR9: DDR2 support not available

Description: Devices: MPC8349E, MPC8347E, MPC8343E

DDR2 support is not available.

Impact: Cannot use DDR2.

Workaround: None

Fix plan: Functionality added to silicon revision 3.x

DDR10: Signals MODT[0:3], MBA2, MDIC[0:1] are not available

Description: Devices: MPC8349E, MPC8347E, MPC8343E

DDR2 support is not available.

Impact: Cannot use DDR2.

Workaround: None

Fix plan: Functionality added to silicon revision 3.x

DDR12: DDR input crossing-point voltage does not meet JEDEC specifications

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The 834x DDR controller may not meet the JEDEC specification (between 1.05 and 1.45 V) for the input crossing-point voltage, VIX(ac).

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

Workaround: No.

Fix plan: Fixed in Rev 3.x.

DDR14: The automatic CAS-to-Preamble feature of the DDR controller can calibrate to incorrect values

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

Fix plan: No plans to fix

I2C1: Enabling I²C could cause I²C bus freeze when other I²C devices communicate

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

General2: Write access to a module with clock disabled causes the device to hang

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Write access to registers of a module with the clock disabled (by programming SCCR register; possible modules are TSEC1, TSEC2, USB-MPH, USB-DR, SEC, SEQ, PCI1, PCI2 or I2C1) causes the device to hang without any error information captured in the arbiter error address and attributes registers.

Impact: Device hangs without capturing any error information in the arbiter error address and attributes register.

Workaround: Do not perform register programming to modules which have been clock-disabled.

Fix plan: Fixed in Rev 3.x

General3: CKSTP/LDP pin function reset value

Description: Devices: MPC8349E, MPC8347E, MPC8343E

CKSTP_IN (input) and LDP[0] (output) functions are muxed on the same device pin. After power-on reset, this pin functions as LDP[0](output), and it can be modified by writing to a register (SICRL). However, to use this pin as CKSTP_IN and connect it to an external device, there might be a contention until the register is programmed.

Impact: CKSTP/LDP cannot be used as CKSTOP_IN unless the boot sequencer is used to program CKSTP/LDP to function as CKSTP_IN before boot.

Workaround: 1. Use IRQ7 pin as CKSTOP_IN

2. Use boot sequencer to program SICRL to the desired value before the CPU is allowed to fetch code.

Fix plan: Fixed in Rev 1.1

General4: I²C boot sequencer may fail if $\overline{\text{HRESET}}$ happens when SDA is driven low

Description: Devices: MPC8349E, MPC8347E, MPC8343E

I²C boot sequencer may not succeed in loading data if HRESET occurs when SDA is driven low by an external slave. The I²C slave does not have reset input and it keeps driving the SDA line. As a result, the boot sequencer cannot generate a START command (SCL high and SDA transition from high to low) and the device hangs.

Impact: Failure to load reset configuration words if load from I²C EEPROM is defined. Failure to load boot sequencer commands if boot sequencer is defined. In this case, the chip will hang.

Workaround: None

Fix plan: There are two scenarios

1. Hard reset configuration word (HRCW) is in EEPROM and is accessed through I²C.
2. HRCW is stored in the local bus Flash. Boot sequencer is enabled.

Scenario 1 fixed in Rev 1.1

Scenario 2 fixed in Rev 3.x

General5: Additional 256K clock cycles of $\overline{\text{HRESET}}$ signal

Description: Devices: MPC8349E, MPC8347E, MPC8343E

New I^2C module in revision 3.x will cause the loading of HRCW to be extended by the additional 256K cycles of PCI_SYNC_IN (in the worst case).

Impact: In revision 3.x and following, HRCW loading time will be extended by an additional 256K PCI_SYNC_IN clock cycles.

Workaround: None

Fix plan: No plans to fix

General6: Override of HRCW word via JTAG when source is the I2C bus may not work if EEPROM on I2C bus contains an incorrect preamble

Description: Devices: MPC8349E, MPC8347E, MPC8343E

JTAG has the ability to override the HRCW word read by the device through writes to internal registers. This procedure should not be dependent on the source of the HRCW word. However, if the source of the HRCW is the I^2C bus and there is an invalid preamble stored in the EEPROM, the override procedure will not work. The I^2C bus will retry the HRCW fetch until it receives a good preamble. This will prevent the override sequence from being successful.

Impact: If loading the HRCW from an EEPROM on the I^2C bus and the EEPROM is blank or has an incorrect preamble, the overriding sequence will not work and the device will hang.

Workaround:

1. Temporarily tie the SDA to the SCL pin rendering the I^2C bus non-functional.
2. Pull up the reset config pins to set the HRCW to load from a different interface such as local bus, then override the HRCW word.
3. Pull up the reset config pins to use one of the default pre-loaded HRCW so the JTAG tool can gain control. Then program the I^2C EEPROM with the appropriate HRCW.

Fix plan: Fixed in Rev 3.x

General12: CSB deadlock

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Case 1: Instruction Caching is disabled for PCI space

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

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

Case 2: Instruction Caching is enabled for PCI space

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

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

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

Workaround: Use one of the following workarounds:

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

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

TSEC5: TSEC in RGMII/RTBI mode does not receive data correctly

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When TSEC (both TSEC1 and TSEC2) operates in RGMII/RTBI mode, it does not receive data correctly.

Impact: TSEC cannot operate in RGMII/RTBI mode.

Workaround: Use GMII/TBI/MII for TSEC.

Fix plan: Fixed in Rev 1.1.

TSEC6: TSEC receiver FIFO overflow can cause the receiver to drop frames or hang (Case 1)

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When the Rx FIFO overflows (a rare condition) due to a system busy condition (that 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), it is possible for the receiver to drop frames without a 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. If the receiver hangs, it cannot recover without chip reset.

Impact: System drops Rx frames or hangs if the Rx FIFO overflows.

Workaround: FIFO overflow should be avoided by adjusting the FIFO emergency thresholds and emergency priority at CSB bus so that when the TSEC enters emergency mode, it gets priority on CSB to allow it to empty the FIFO at a faster rate than the receive rate.

Fix plan: Fixed in Rev 1.1.

TSEC6.1: TSEC receiver FIFO overflow can cause the receiver to drop frames or hang (Case 2)

Description: Devices: MPC8349E, MPC8347E, MPC8343E

It is possible for the TSEC'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 TSEC. It may take longer for the TSEC 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 that 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 will bring the TSEC out of this condition.

Impact: System drops Rx frames or hangs if the Rx FIFO overflow happens.

Workaround: FIFO overflow should be avoided by adjusting the FIFO emergency thresholds and emergency priority at the CSB so when the TSEC enters emergency mode, it gets priority on the CSB empty the FIFO at a faster rate than the receive rate. Flow control should be enabled whenever possible. SPCR, ACR, SCCR and Rx FIFO must be set as follows to avoid the Rx FIFO overflow condition:

1. SPCR = 0x0000_0707. OPT and TBEN bits can be set according to need.
2. ACR[PCI_RPTCNT]=0b000, ACR[RPTCNT]=0b000, ACR[PIPE_DEP]=0b011.
3. SCCR[TSEC1CM]=0b01, SCCR[TSEC2CM]=0b01.
4. Rx FIFO must be set as follows. The address of the register is the corresponding offset plus the TSEC1 and TSEC2 base address, which are 0x2_4000 and 0x2_5000, respectively.

RX_FIFO_ALARM: offset 0x050, default is 0x100, set to 0x080.

RX_FIFO_ALARM_SHUTOFF: offset 0x054, default is 0x080, set to 0x040.

RX_FIFO_PANIC: offset 0x058, default is 0x180, set to 0x100.

RX_FIFO_PANIC_SHUTOFF: offset 0x05C, default is 0x100, set to 0x080.

Fix plan: No plans to fix

TSEC7: RGMII timing does not meet with RGMII specification

Description: Devices: MPC8349E, MPC8347E, MPC8343E

In the RGMII specification, t_{SKRGT} = –0.5 (minimum) to 0.5 (maximum) ns. The correct result from device characterization: t_{SKRGT} = –1.0 (minimum) to 0 (maximum) ns.

Impact: No.

Workaround: 1. Use a software workaround (when working in RGMII mode) by programming the I/O output delay register (currently hidden to users) at IMMR offset 0x120 to put t_{SKRGT} back into the 0 to 1.0 ns range.

The register value should be ANDed with 0xFFFE_001F:

(long) (IMMRBAR + 0x120) &= 0xFFFE_001F;

2. Use a hardware workaround by adding, in the board, at least 1.0 ns delay to the TX clock signal to match the spec at PHY input.

Fix plan: No plans to fix

TSEC20: TBI transmit AC timing specification tTTKHDX is not met

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The TBI transmit AC timing specification TCG hold time tTTKHDX of 1.0 ns minimum is not met. The hold time measures as little as 700 ps.

Impact: If the interacting PHY does not have the margin to accommodate the timing degradation, incorrect data may be latched by the PHY.

Workaround: When working in TBI mode, use the reserved register at offset IMMRBAR + 0x0120 to add delay to the TBI outputs. Change bits 14–19 to 0b101010 for adding delay to TSEC1 outputs. Change bits 20-25 to 0b101010 for adding delay to TSEC2 outputs. Preserve the default value of the other bits. The additional delay that will be added by this programming is in the range of 0.5ns to 1.2ns.

Fix plan: Fixed in Rev 3.x.

TSEC21: The MPC834x does not meet the TSEC GMII input hold timing

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The TSEC GMII input hold timing, as stated in the device hardware specifications, is not met under all conditions. The minimum value of t_{GRDXKH} (data hold time to CLK) is stated to be 0.5 ns. GMII inputs may require an additional 400 ps of hold time.

Impact: If the GMII inputs hold are not increased the TSEC could latch wrong data.

Workaround: Add hold time on GMII inputs at board level. Add at least 400 ps but no more than 1ns.

Fix plan: Fixed in Rev 3.x..

TSEC22: The power supply of TSEC1, LVDD1, is internally tied to the power supply of TSEC2, LVDD2

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The power supply of TSEC1 (LVDD1) is internally tied to the power supply of TSEC2 (LVDD2). As a result, TSEC1's power supply must be set to the same voltage as TSEC2's power supply.

Impact: TSEC1 and TSEC2 must operate at the same voltage. For example, if TSEC1 is using GMII, TBI, or MII (3.3 V), then TSEC2 cannot run RGMII or RTBI (2.5 V). In addition, some of GPIO pins cannot go up to 3.3 V when either LVDD1 or LVDD2 power supply is set to 2.5 V. For more information, see GPIO1.

Workaround: Do not use GMII, TBI, or MII on one TSEC and RGMII or RTBI on the other TSEC; use the same power supply for TSEC1 and TSEC2 by setting LVDD1 and LVDD2 to either 2.5 V or 3.3 V. Do not use GPIO pins that are muxed with TSEC pins when LVDD1 and LVDD2 are set to 2.5 V.

Fix plan: Fixed in Rev 3.x.

PCI2: PCI returns bad data on a master read following perr_response assertion

Description: Devices: MPC8349E, MPC8347E, MPC8343E

If the PERR bit of the PCI bus command register (0x04 in the configuration space) is changed from 0 to 1 by using the CFG_DATA register, and immediately following this change there is a master read from PCI space, the wrong read data is returned to the CSB initiator. This only occurs if the CSB/PCI clock ratio is higher than 6:1.

Impact: Bad read data could lead to total system failure if it contains instructions.

Workaround: Writing to the register twice or writing to and then reading from the register prevents the situation from occurring.

Fix plan: Fixed in Rev 3.x

PCI3: Data corruption when bus arbiter is parked to PCI and external PCI master performs transactions with non-consecutive byte enables

Description: Devices: MPC8349E, MPC8347E, MPC8343E

If the bus arbiter is parked to PCI, the non-consecutive byte enable transactions (for example, $\overline{CBE}$= 0101) from the external master cannot be completed correctly.

Impact: Workaround has small performance degradation compared to PCI as a parked master.

Workaround: For the system with an external PCI master that could generate non-consecutive byte enable transactions, do not park the CSB bus to PCI.

Fix plan: Fixed in Rev 3.x

PCI4: Data corruption if PCI is accessed as CSB slave device and CPU snoops

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When PCI is accessed as CSB slave and the memory space is global, then CPU core may retry the transaction. The retry logic causes data corruption under this scenario.

Impact: The retry logic causes data corruption under this scenario.

Workaround: PCI outbound memory space must be defined as non-global. If the CSB masters, such as TSEC/USB/Security, need to access PCI outbound memory space, the memory space must be defined as non-global in the corresponding masters' control registers. ATTR for TSEC, SNOOP1/2 for USB and MCR for Security block.

Fix plan: Fixed in Rev 3.x

PCI5: Target read transactions of more than one cache line from prefetchable memory space may cause data corruption

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When external PCI devices try to read from the memory which is defined as prefetchable and where the transaction is more than one cache line, there may be data corruption.

Impact: Workaround impacts the PCI inbound performance.

Workaround: 1. Use the memory read multiple PCI command for any burst that crosses the cache line boundary.

or

2. Do not mark any inbound windows to be prefetchable. This likely causes a significant reduction in the bandwidth of target reads.

or

3. The master should disconnect at every cache line boundary. This may cause a reduction of up to 30% in the bandwidth of target reads as compared to long bursts.

or

4. Using on-chip DMA to transfer data.

Fix plan: Fixed in Rev 1.1

PCI6: PCI host configuration read from unused slot terminates with CSB error

Description: Devices: MPC8349E, MPC8347E, MPC8343E

PCI host configuration read from unused slot terminates with CSB error. Normal operation requires that reading from unused slots returns FFFF_FFFF and no error is reported. CSB error is translated to machine check interrupt when the transaction initiator is the CPU.

Impact: Auto-configuration process cannot be performed.

Workaround: None

Fix plan: Fixed in Rev 1.1

PCI7: PCI subsystem ID registers are writable from the PCI side

Description: Devices: MPC8349E, MPC8347E, MPC8343E

PCI subsystem ID registers should be read-only from PCI side. In practice, they are writable.

Impact: Software should avoid a write to the registers.

Workaround: None

Fix plan: Fixed in Rev 1.1

PCI8: CFG_CLKIN_DIV cannot be set high in PCI agent mode

Description: Devices: MPC8349E, MPC8347E, MPC8343E

CFG_CLKIN_DIV cannot be set high in PCI agent mode. If CFG_CLKIN_DIV is set high at power-on reset, the PCI controller will not be synchronized to the PCI clock (PCI_SYNC_IN). As a result, CSB frequency cannot be kept running at the same frequency when M66EN is detected low.

Impact: Clock cannot run in this mode.

Workaround: None

Fix plan: Fixed in Rev 1.1

PCI9: The device can hang if PCI inbound transaction or DMA transaction with snoop enabled happens at the same time as the CPU tries to access PCI outbound window

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The device can hang if PCI inbound transaction or DMA transaction with snoop enabled happens at the same time as the CPU tries to access PCI outbound window, and the CPU is programmed as the parked master.

Impact: Device hangs under the described condition.

Workaround:

- Disable CSB parking or park to PCI.

or

- Turn off snoop bit in DMA and PCI inbound.

Fix plan: Fixed in Rev 3.x.

PCI10: PCI inbound write data may be corrupted if a CSB/PCI clock ratio of 2:1 is used

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When using a CSB/PCI clock ratio of 2:1, PCI inbound write data may be corrupted.

Impact: CSB/PCI 2:1 clock ratio cannot be used.

Workaround: Use a higher clock ratio.

Fix plan: Fixed in Rev 3.x.

PCI11: Sub class ID setting is wrong in the host mode

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The class code register values do not match the values in the device reference manual, Rev. 1. The class code register should return a value of 0x0b2000. In a device to which this errata is applicable, the subclass value will be 0x068000 for host mode and 0x060000 for agent mode.

Impact: Causes confusion

Workaround: No.

Fix plan: Fixed in Rev 3.x.

PCI12: PCI Rev 2.3 is not available

Description: Devices: MPC8349E, MPC8347E, MPC8343E

PCI Rev 2.3 support is not available. (PCI Rev 2.2 is available in silicon revision 1.0.)

Impact: Cannot use PCI Rev 2.3.

Workaround: None

Fix plan: Functionality added to silicon revision 3.x

PCI13: Data corruption occurs when an external PCI initiator issues the “Read Multiple” command

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When the 83xx PCI controller is the target of a PCI Memory Multiple-Read (MRdMult) command, the IO sequencer may enter an illegal state, causing data corruption on the PCI bus and other erroneous behaviors.

Impact: When the IOS buffer control mechanism becomes out of sync, the following behaviors may occur:

- Data corruption on the PCI bus
- System-level data corruption
- Unexpected PCI transactions
- Missed PCI transactions

Workaround: Use one of the following options:

- Turn off the MRdMult command on any potential external PCI initiator.
- Limit the DMA transaction size to a cache line 32 Bytes or less when using the MRdMult command on the PCI initiator side.
- Do not mark any inbound windows to be prefetchable. This will likely impact performance due to significant reduction in the bandwidth of PCI target reads.

Fix plan: Fixed in Rev 3.x.

PCI14: PCI input setup time will be changed to meet PCI AC timing specification

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The PCI input setup time requirement, t_{pcivkh}, operating at 33 MHz will be changed from a minimum of 3 ns in the HW Spec to 7 ns to match the JEDEC PCI AC timing specification. The minimum PCI input setup time for 66 MHz will remain unchanged at 3 ns.

Impact: The PCI AC input setup time for 33 MHz has been tightened.

Workaround: No

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

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

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

Workaround: Hardware workaround:

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

Software workarounds:

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

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

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

Workaround:

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

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

The sequence of the failure is as follows:

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

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

Workaround: Use one of the following options:

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

Fix plan: No plans to fix

LBC1: UPM does not have indication of completion of a RUN PATTERN special operation

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

The following scenario could be affected by this erratum:

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

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

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

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

Workaround: None

Fix plan: No plans to fix

LBC2: LBC clock glitch when performing synchronous accesses with DLL enabled

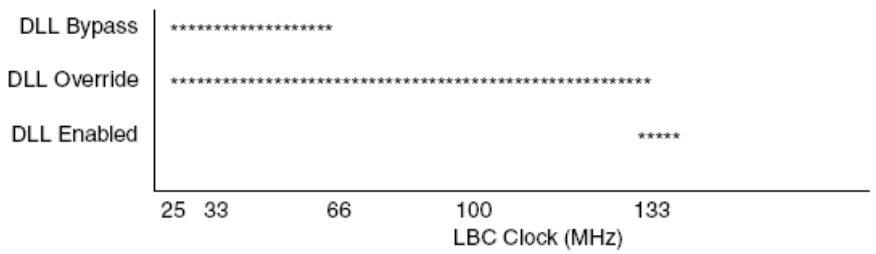
Description: Devices: MPC8349E, MPC8347E, MPC8343E

An infrequent irregularity occurs on the local bus interface when performing synchronous accesses with the LBC DLL enabled. The issue appears as a local bus clock glitch which can result in data corruption during read or write cycles. This issue does not present itself when the local bus is operating at greater than 133 MHz with the DLL enabled.

The local bus is designed to operate in two modes: DLL bypass and DLL enabled (as configured by the LCRR[DBYP] bit).

Impact: A local bus clock glitch which can result in data corruption during read or write cycles.

Workaround: This workaround (DLL override) is applicable for the device in DLL-enabled mode only. The recommended operating ranges for each of the LBC modes are as follows:



The following sequence of events is required at device boot-up to manually override the DLL:

1. Wait for $\overline{\text{HRESET}}$ to negate.
2. Program the LBC configuration registers accordingly, especially LCRR, setting LCRR[DBYP] = 0.
3. Wait for 100 microseconds (for example, in a loop), minimizing core activity.
4. Perform the following code sequence:

```
(uint)x = DLLSR; // Read measured tap from local-bus DLL status
x |= 0x80000000; // set override bit
DLLOVR = x;      // Write override tap point to local-bus DLL override
reg
                 // and set override bit
```

Fix plan: No plans to fix

LBC3: Chip select pins 4–7 not available (LCS[4:7])

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Supports only four banks of memory.

Impact: Can only use $\overline{\text{LCS}}$[0:3].

Workaround: None

Fix plan: Functionality added to silicon revision 3.x

LBC5: $\overline{\text{LCS}}$[6:7]/PCI_CLK[3:4] drive zeros during reset

Description: Devices: MPC8349E, MPC8347E, MPC8343E

PCI_CLK[3:4]/$\overline{\text{LCS}}$[6:7] pins drive zeros during and after the period in which $\overline{\text{PORESET}}$ and $\overline{\text{HRESET}}$ are asserted. If a slave device is connected to PCI_CLK[3]/$\overline{\text{LCS}}$[6] or PCI_CLK[4]/$\overline{\text{LCS}}$[7] and this slave will drive data on LAD signals when it has $\overline{\text{LCS}}$ and $\overline{\text{LOE}}$ both asserted, this will lead to contention on the LAD bus and errors in loading the RCW data. It will also lead to errors when the e300 CPU will try to fetch boot data.

Impact: Cannot load RCW from local bus. Cannot boot from local bus.

Workaround: Use one of these alternatives:

- Use I^2C EEPROM to load RCW, and use the boot sequencer to set SICRH[27] before letting the CPU boot its code.
- If applicable, use a hard coded RCW and use the remote PCI host to set SICRH[27] before letting the CPU boot its code.
- If applicable, use LCS[6:7], which are multiplexed with SPI pins.
- Do not connect local bus slaves to LCS[6:7].

Fix plan: No plans to fix

SPI1: Third character received causes overflow if the core does not read first and second characters

Description: Devices: MPC8349E, MPC8347E, MPC8343E

SPI receiver supports a fifo of two characters. If a third character will be received without reading the receive fifo, an overflow will occur although the OV bit will not be set.

Impact: Data is corrupted.

Workaround: Handle the receiver at least once (if interrupt is asserted) after no more than 2 characters are received.

Fix plan: Fixed in Rev 3.x.

SPI2: SPIE[NF] is set after transmission of a one byte block

Description: Devices: MPC8349E, MPC8347E, MPC8343E

If the number of bytes to be transferred is one (that is, the Last command (SPCOM[LST]) is set for the first character), the SPI asserts the NF (Not Full) status bit (SPIE[NF]) after writing this byte to the transmitter, although it should not. The Last command should clear the NF (Not Full) status bit in the SPI Event Register (SPIE) until the next character of a new block is written. In this case (first byte is also the last) the NF bit will still be set after the write of the last character.

Impact: The CPU gets another request for data.

Workaround: CPU can ignore the additional request (that is, mask the interrupt).

Fix plan: Fixed in Rev 3.x.

SPI3: SPIE[NF] bit not set after writing the first character

Description: Devices: MPC8349E, MPC8347E, MPC8343E

In SPI master mode, if the last character in a transmitted block of characters (SPCOM[LST] is set for this character) is written to the transmitter when the transmitter is idle (that is, it has completed the transmission of the previous data), the SPIE[NF] status bit will not be set after writing the next character (the first character in a new block).

Impact: The CPU will not get additional request for data after writing the first character to the SPI transmitter.

Workaround: In SPI Master mode, do not use the last command, ignore/mask the NF indication after transmission of the last character. Alternatively, disable and re-enable the SPI after transmission.

Fix plan: Fixed in Rev 3.x.

SPI4: LT bit in SPI event register is never set

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The LT bit in the SPI event register should be set by hardware as a result of software setting the LST bit in the SPI command register and when the transmission of the last character has ended. In device revisions to which SPI4 applies, LT is never set; as a result, the software is not able to determine when transmission of the last character has ended and it is safe to negate slave select output.

Impact: No interrupt happens, in infinite waiting.

Workaround: The LST bit in the SPI command register will be cleared by hardware when the transmission of the last character has ended. Software can monitor SPICOM[LST] for this purpose. This workaround will not be valid in future revisions.

Fix plan: Fixed in Rev 3.x.

USB1: USB data corruption after read burst in which system bus error was generated

Description: Devices: MPC8349E, MPC8347E, MPC8343E

USB may corrupt DMA read data for a subsequent single-beat transaction after a read burst in which a system bus error is generated. This corruption occurs regardless of the value of the ‘err_disable’ bit.

Impact: If the single-beat read that follows the errored burst is USB data, then the corresponding data on the USB is corrupted. If the single-beat read that follows the errored burst is USB control information, then invalid USB controller operation can result. If the errored read burst is also USB control information, invalid USB controller operation is likely to occur regardless of the subsequent corrupted single-beat read.

Workaround: There is no work-around. Host or device software should leave the ‘err_disable’ bit negated. It is recommended that system software configure the system bus arbiter to generate an interrupt to the host whenever address bus timeouts occur.

Fix plan: Fixed in Rev 3.x.

USB3: MPH USB port 1 will not operate without MPH0_CLK signal present on port 0

Description: Devices: MPC8349E, MPC8347E

When the device is configured for the multi-port host controller, USB port 1 (MPH port 1) is not set correctly if a clock is not present on the MPH0_CLK pin. MPH1_D7_DRVVBUS remains in input mode.

Impact: HS ULPI Mode:

This condition will only occur for system configurations that do not have a ULPI PHY attached to USB Port 0 and the device is configured to use the multi-port host controller. For example, a single port host implementation which only utilizes MPH1 of the MPH controller.

FS/LS Serial Mode

This condition will only occur for system configurations that do not have the 60 MHz serial clock attached to MPH0_CLK and the device is configured to use the multi-port host controller. For example, a single port host implementation which only utilizes MPH1 of the MPH controller.

Workaround: HS ULPI Mode:

For system configurations that do not have a ULPI PHY connected to USB PORT 0, tie the MPH0_CLK pin to the MPH1_CLK pin.

FS/LS Serial Mode:

Always tie the MPH0_CLK pin to the MPH1_CLK pin.

Fix plan: Fixed in Rev 1.1

USB4: ULPI interface does not recognize rx_active immediate

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The immediate form of rx_active is not properly decoded. This affects ULPI operation only. rx_active is used to signal the end/start of packets. The assertion of rx_active is used to signal the start of a packet. rx_active can be signaled via the ULPI rx command or by the assertion of the NXT and DIR signals. The simultaneous assertion of DIR and NXT (rx_active immediate) should be interpreted as rx_active asserted. The current implementation does not decode the rx_active immediate condition. If the PHY uses rx_active immediate signaling, the link will not recognize the PID following the rx_active assertion. This will cause USB protocol errors and halt the controller. The device will not be able to receive packet correctly.

Impact: The device USB will not work with the PHY that takes advantage of the rx_active_immediate signaling on the ULPI.

Workaround: No.

Fix plan: Fixed in Rev 1.1.

USB5: RX EOP Detection Error

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The ULPI allows the PHY to signal the end of a receive packet by setting the rx_active bit low in an rx command or by de-asserting the DIR signal. The current implementation does not recognize de-assertion DIR as an end of packet.

Impact: This affects ULPI mode only. If a PHY uses DIR to signal an end of packet, the USB controller will not recognize the end of the receive packet, and CRC and packet length violations will be generated.

Workaround: No.

Fix plan: Fixed in Rev 1.1.

USB6: Transmit to transmit inter-packet gap

Description: Devices: MPC8349E, MPC8347E, MPC8343E

USB specification states that back-to-back transmit packets must have an inter-packet delay of between 88–192 bit times. In ULPI mode, the HS inter-packet delay should be timed from the assertion of the STP signal. The current implementation begins timing the HS inter-packet delay from the recognition of EOP on the line state signals.

Impact: This affects HS ULPI only. Timing from STP accommodates for the delay from the assertion of STP and the PHY signaling EOP on the USB. ULPI specifies the PHY must signal EOP on the bus 2–5 bit times after the STP is asserted. However, there is no specification for the time the PHY will present the EOP state on the linestate signals that the link is monitoring. The device USB controller inter-packet delay is set to 160 bit times, but because the timer does not start until the EOP is observed, the delay between STP assertion and EOP must be added to the delay. This delay may take an extra clock cycle or two from when the EOP is placed on the USB. This would extend the inter-packet delay to 216 bit times, which is outside the USB spec. This should not affect operation of the USB because this is still well below the USB HS bus time-out specification of 736 bit times.

Workaround: No.

Fix plan: Fixed in Rev 1.1.

USB7: ULPI Register read commands corrupt the receive packet

Description: Devices: MPC8349E, MPC8347E, MPC8343E

ULPI register read commands executed through the ULPI viewport that are initiated immediately before a receive packet can corrupt the receive packet.

Impact: This affects ULPI mode operation in both MPH and DR controllers. The impact is considered minimal because the viewport access is not intended for use during normal operation while USB packets are in flight.

Workaround: The user should not attempt to perform ULPI viewport register reads while USB packets are in flight.

Fix plan: Fixed in Rev 3.x.

USB8: Early LS Handshake Time-out

Description: Devices: MPC8349E, MPC8347E, MPC8343E

If an expected handshake packet is not received within 16–18 bit times, a bus time-out should occur. Presently, a LS handshake is rejected and a bus time-out occurs if the handshake is received after ~13 bit times.

Impact: This affects LS host operation only. After the host sends a data packet, the device acknowledges the receipt of the data with a handshake token (ACK). At the completion of sending a data packet, the host waits for the handshake from the device. The device must begin sending the handshake within 7.5 bit times. In a worst case USB topology (max. cable length and 5 hubs w/max. delay) with the max. 7.5 bit time turnaround time, the handshake would arrive ~8 bit times after completion of the data packet transaction. This is well within the time-out period of 13 bit times. This should not affect operation of the USB but does violate the USB spec.

Workaround: No.

Fix plan: Fixed in Rev 3.x.

USB9: FS EOP insertion on pre-PID

Description: Devices: MPC8349E, MPC8347E, MPC8343E

A FS EOP is inserted after a pre-PID in FS/LS serial mode. This is not a valid configuration for a pre-PID transaction.

Impact: This affects FS serial host operation only. This will only be observed when an external hub operating in FS mode has a LS device attached. When an external hub is attached to the USB and a low speed device is attached to a downstream port, a pre-PID token is used to inform the hub to enable the low speed port. The normal transaction would include the pre-PID followed immediately by the sync pattern of the next transaction. The host controller presently inserts an EOP before the sync pattern. This error has never been observed to cause a problem during any inter-operability testing.

Workaround: Do not connect LS speed device through an external hub when in FS serial mode.

Fix plan: Fixed in Rev 3.x.

USB10: Test J/Test K mode configuration

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Entering Test J or Test K modes does not disable bit stuffing in the ULPI PHY. When USB controllers are programmed for Test J and Test K modes, bit stuffing must be disabled in the PHY. This should be accomplished automatically by hardware through a ULPI register write command to the function register in the ULPI PHY. The current implementation does not issue the ULPI register write command.

Impact: This affects ULPI operation only. Test J and Test K modes are used to facilitate compliance testing. They are not used for normal operation. When using Test J and Test K modes, bit stuffing must be disabled by software prior to entering the test modes.

Workaround: Software must disable bit stuffing in the ULPI PHY by issuing a ULPI register write command to the ULPI functional control register through the ULPI viewport prior to setting the test modes (PTC[3:0]) in the PORTSCx register.

Fix plan: Fixed in Rev 3.x.

USB11: Low power resume wait delay

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Resume signaling can be missed by the USB controllers because the 5 ms suspend-to-resume timer has not expired prior to receiving a resume signaling.

Impact: The USB specification requires a 5 ms delay between entering suspend state and issuing a resume signaling. USB controllers have built in timers to ensure this delay. However, if the link signals the PHY to enter low power mode due to suspend, the PHY will shut down the PHY clocks. This prevents the 5 ms delay timer from continuing, thus causing the link to be held between the suspend and resume states, preventing the link from recognizing resume signaling. This is not an issue for applications that will not shut down the PHY clocks during low power mode.

Workaround: When software recognizes a suspend condition, it must wait 5 ms before putting the PHYs into low power suspend state.

Fix plan: Fixed in Rev 3.x.

USB12: Stall bit setting during setup lockout

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The USB DR controller contains a lockout and semaphore bit to prevent setup data from being corrupted while the device controller driver (DCD) is servicing setup transactions. If the DCD stalls packets that are out of band of the normal setup sequence, a race condition can arise that could allow the setup data to be corrupted.

Impact: This errata affects the DR controller in device mode only. This errata only appears when attempting an abnormal setup sequencing.

Workaround: The DCD should not write to the endpoint stall bit while in the setup lockout period. The setup lockout period is timed from the reception of a setup token to the time the DCD completes the servicing of the setup command.

Fix plan: Fixed in Rev 3.x.

USB13: Host lock up on the disconnect

Description: Devices: MPC8349E, MPC8347E, MPC8343E

If an FS port is disconnected during back-to-back transmit transactions, that is, OUT token followed by DATA, the host controller can lock up. The only recovery option is a hard or soft reset of the affected USB controller.

Impact: The errata only affects FS, ULPI, or Host operation, while performing back-to-back transmit operations to bulk, interrupt, or control endpoints. Isochronous transactions are not affected. If this scenario occurs the USB controller must be reset.

Workaround: Whenever a disconnect event is detected, through the port change interrupt, if the USB controller is operating in Host mode at FS with a ULPI interface, software should generate a soft reset by halting the USB controller and then setting the RST bit in the USBCMD register.

The following pseudo code illustrates this workaround:

```
if (host and FS and ULPI and disconnect)
 then
 USBCMD[RUN/STP] = 0
 wait (USBSTS[HCHalted] == 1)
 USBCMD[RST] = 1
```

Fix plan: Fixed in Rev 3.x.

USB14: Port numbering in Queue Head violates EHCI specification

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Port number in the Queue head is 0 to N – 1. It should be 1 to N according to EHCI spec. This bug only affects the host mode.

Impact: Standard USB stack does not work with the MPC8349 host controller.

Workaround: Use the port number 0 to N – 1 instead of 1 to N.

Fix plan: Fixed in Rev 3.x.

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

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

Fix plan: No plans to fix

USB17: USBDR and MPH do not generate interrupt upon CRC16 error in IN data packet when in host mode.

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When a CRC16 / PID error is present, the USBDR and MPH correctly detect the error and modify the appropriate error status bit in the data structures. However, they neither generate an interrupt nor set the corresponding bit in USBSTS[UEI] register. NOTE: The interrupt is generated properly if an EOP error is present.

Impact: The impact is not critical, because the appropriate error bits in the asynchronous / synchronous list are correctly flagged, so software can work around this issue.

Workaround: Interrupt polling should be implemented in the customer's software driver. The software driver should poll the error information already written in the status field of the data structure.

Fix plan: No plans to fix

USB21: SE0_NAK issue

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When put into SE0_NAK test mode in device configuration, the USB controller may occasionally miss an IN token (not respond with a NAK token), if it was issued exactly on 125 microsec micro-frame boundary, when SOF is expected in functional mode.

Impact: None

Workaround: None

Fix plan: No plans to fix

USB22: USB clock to output valid allows less margin on input setup time

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The USB clock to output valid (t_{USKHOV}) for all outputs is specified for 7 ns for the maximum value. The correct specification from device characterization is 8 ns.

Impact: Less margin on input setup time on the USB PHY side.

Workaround: None

Fix plan: No plans to fix

USB24: A write to some USB registers such as PORTSC can cause the system to hang when no USB PHY clock is applied

Description:

NOTE

This erratum is not applicable to MPH configuration, and only applies when DR configuration is used.

Devices: MPC8349E, MPC8347E, MPC8343E

The issue is that some of USB registers are clocked by the PHY clock, and not by the platform clock. During the write access, there are handshake signals between the two clock domains which indicate end of transaction. The system cannot finish the handshake if no USB PHY clock is applied. There should not be problems with read accesses even if the PHY is shut down. The related USB registers are as follows:

- Offset 0x154—DEVICE_ADDR/PERIODICLISTBASE
- Offset 0x1c0—ENDPTCTRL0
- Offset 0x1c4—ENDPTCTRL1
- Offset 0x1c8—ENDPTCTRL2
- Offset 0x1cc—ENDPTCTRL3
- Offset 0x1d0—ENDPTCTRL4
- Offset 0x1d4—ENDPTCTRL5
- Offset 0x184—PORTSC

Keep in mind that the problem occurs only if these registers are written when PORTSC[PHCD] = 0. There is no issue when PORTSC[PHCD] = 1. Performing these register accesses when PORTSC[PHCD] = 0 and no PHY clock is present is out of our specification. The definition of PORTSC[PHCD] in the reference manual already states “Note: If there is no clock connected to the USBn_CLK signals, PHCD must be set.” Generally, there's no reason to write to these registers if there is no USB clock, unless due to a programmer's error.

Impact: When no USB PHY clock is applied, a software writing to some USB registers, such as PORTSC, can cause the system bus access hang.

Workaround: Do not write to USB registers if no USB PHY clock is present.

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

Fix plan: No plans to fix

USB31: Transmit data loss based on bus latency

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When in Host mode, if an Rx FIFO overflow happens close to the next Start-of-Frame (SOF) token and the system bus is not available, a false frame babble is reported to software and the port is halted by hardware. If one SOF is missed, the Host controller will issue false babble detection and SOFs will no longer be sent. If more than 3.125 ms are elapsed without SOFs, the peripheral will recognize the idle bus as a USB reset.

Impact: If this scenario occurs, it will degrade performance and have system implications. The Host will have to reset the bus and re-enumerate the connected device(s).

Workaround: Reset the port, do not disable the port, on which the babble is detected.

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

Fix plan: No plans to fix

USB34: NAK counter decremented after receiving a NYET from device

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

Impact: The NAK counter may be lower than expected.

Workaround: None

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

In the case where the Controller is configured as a device and the Host sends two consecutive ISO OUT (example sequence: OUT - DATA0 - OUT - DATA1) transactions with a short inter-packet delay between DATA0 and the second OUT (less than 200 ns), the device will see the DATA1 packet as a short-packet even if it is correctly formed. This will terminate the transfer from the device's point -of-view, generating an IOC interrupt. However, DATA0 is correctly received.

Impact: If this scenario occurs, the clear command from the protocol engine state machine to the protocol engine data path is not sent (and internal byte count in the data path module is not cleared). This causes a short packet to be reported to the DMA engine, which finishes the transfer and the current dTD is retired.

Workaround: None

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

In systems with high latency, the HOST can under-run on OUT transactions. In this situation, it is expected that the CRC of the truncated data packet to be the inverted (complemented), signaling an under-run situation.

Impact: Due to this erratum, the controller will not send this inverted CRC. Instead, it sends only one byte of the inverted CRC and the last byte of payload.

It is unlikely but remotely possible that this sent CRC to be correct for the truncated data packet and the device accepts the truncated packet from the host.

Workaround: Setting bigger threshold on TXFILLTUNING[TXFIFOTHRES] register might solve the under-run possibility and thus avoiding truncated packets without the expected inverted CRC.

However, this would not solve the inverted CRC issue by itself.

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

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

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

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

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

Fix plan: No plans to fix

USB38: USB Controller locks after Test mode "Test_K" is completed

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

Workaround: None

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

Impact: Two different situations may occur:

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

Workaround: None

Fix plan: No plans to fix

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

Description: Devices: MPC8343E, MPC8347E, MPC8349E

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

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

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

Impact: There are two cases to consider:

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

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

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

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

Workaround:

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

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

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

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

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

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

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

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

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

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

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

Workaround: None

Fix plan: No plans to fix

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

Workaround: None

Fix plan: No plans to fix

SEC1: Global soft reset does not work when interrupts are pending

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The SEC 2.0 is designed to be globally reset by either a full reset or by a write to the SEC's master control register global software reset bit. If there is an error pending in one or more SEC crypto-channels and the SEC is reset through a write to the SEC's master control register global software reset bit, the crypto-channel(s) with the pending errors are not properly reset. The rest of the SEC, upon completion of reset will be ready for operation, however the affected crypto-channel(s) will not execute new fetch commands until the interrupt is cleared.

Impact: None.

Workaround: This errata does not affect hard resets, only attempts to reset the SEC through a single write to the SEC's master control register global software reset bit. The work-around is to perform two back-to-back writes to the SEC's master control register global software reset bit. The SEC 2.0 reference device driver includes this work-around. The double write to the SEC master control register global software reset bit does not impact performance, as a full reset of the SEC would typically only be performed during debug, or in the event of significant non-recoverable errors affecting security processing.

Fix plan: Fixed in Rev 3.x.

SEC2: Public key execution unit does not work

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Public key execution unit does not work.

Impact: Public key execution unit cannot be used.

Workaround: None.

Fix plan: Fixed in Rev 1.1.

SEC3: AESU Input FIFO error on less than 16B data size

Description: Devices: MPC8349E, MPC8347E, MPC8343E

There is a logical race condition between how the AESA input FIFO handles a last incomplete word written, and the starting of processing of the last block of data. When AESA is processing a packet in CCM mode, and that packet is greater than 8 bytes in length and less than 16 bytes in length, the race condition causes the incomplete half-block to be ignored. The partial block gets left in the FIFO and causes an ERROR to be asserted.

Impact: High in 802.16 applications.

Workaround: Software must check the data size, and if the data is less than 16 bytes, perform the following steps:

For encryption:

Utilizing the SEC's scatter-gather capability, substitute a link table pointer for the pointer to the plaintext payload. The Link Table should contain two pointers: a pointer to the payload (length is the actual plaintext data length) followed by a pointer to a number of bytes of zeroes in a previous prepared memory region. The purpose of the zeroes is to make the total data size fetched =16B, so the number of required bytes of the zeroes for padding is 16-(plaintext payload length). The length in the main descriptor will be the total of the plaintext data + the zero padding. When outputting the ciphertext, it is not necessary to output the padded length. The ciphertext-out data length will be the same length as the original plaintext.

For decryption, the process is similar, but calculating the padding requires an additional descriptor.

First descriptor: Using AES-CTR mode, encrypt a payload of 16 bytes using the AES-CCM encryption key, and a nonce which is the IV portion of the context with five msbits of the flag byte set to zero and least significant 2 bytes set to the integer value of 1. The result will be a non-zero buffer of 16 bytes.

Second descriptor: Utilizing the SEC's scatter-gather capability, substitute a link table pointer for the pointer to the ciphertext payload. The Link Table should contain two pointers: a pointer to the payload (length is the actual ciphertext data length) followed by a pointer to a number of bytes of the non-zero padding generated by the first descriptor. The purpose of the padding is to make the total data size fetched =16B, so the number of required bytes of the non-zero padding is 16-(plaintext payload length). These padding bytes should be considered the completion of the 16-byte block, so if the ciphertext is 13 bytes, the link table pointer to the padding should point to the last 3 bytes of the non-zero buffer. The length in the main descriptor will be the total of the ciphertext data + the non-zero padding. When outputting the plaintext, it is not necessary to output the padded length. The plaintext-out data length will be the same length as the original ciphertext.

Fix plan: Fixed in Rev 3.x.

SEC4: XOR parity generation accelerator for RAID applications

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The AESU XOR Execution Unit is not available for use.

Impact: Cannot use the XOR parity generation accelerator.

Workaround: None

Fix plan: Functionality added to silicon revision 3.x

SEC5: SHA-224 support is unavailable

Description: Devices: MPC8349E, MPC8347E, MPC8343E

SHA-224 cryptographic hash function is unavailable.

Impact: Cannot use the SHA-224 hash function.

Workaround: None

Fix plan: Functionality added to silicon revision 3.x

SEC6: AES-CTR mode data size error

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

Impact: This work around will impact performance through the additional software overhead of creating two descriptors, and through increased bus consumption, as the HMAC-only descriptor will read the data a second time.

Workaround: Use one of the following options:

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

Fix plan: No plans to fix

SEC7: Single descriptor SRTP error

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

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

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

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

Fix plan: No plans to fix

SEC-A001: Channel Hang with Zero Length Data

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

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

Fix plan: No plans to fix

JTAG1: Under JTAG mode, the pin type of PCI1_INTA and MCP_OUT is wrong

Description: Devices: MPC8349E, MPC8347E, MPC8343E

PCI1_INTA and MCP_OUT are open drain output pins. As such, they are required to drive 0 or Z. However, under JTAG mode, they are not open drain and can drive either 0 or 1. In revision 1.1, it is partially fixed. They can drive 0, 1, or Z.

Impact:

1. Board connectivity test tool cannot put the pin in Z. Logic 1 caused by pull-up resistor cannot be verified.
2. Board connectivity test tool must be aware of contention risk.

Workaround:

1. Board connectivity test tool needs to ensure that these pins will not drive 1.
2. Use the HIGH Z command to put all pins in high Z when PCI1_INTA or MCP_OUT need to be put in high Z. Although these two pins cannot be put in high Z individually, they can go to high Z when whole chip goes high Z via HIGH Z command.

Fix plan: Fixed in Rev 3.x

JTAG2: Under JTAG mode, pin type of certain pins is wrong

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The following pins are listed as output2 in the BSDL file but they should be listed as output3:

MA [0:14]

MBA [0:2]

MCAS_B

MCS_B [0:3]

MRAS_B

MWE_B

PCI1_GNT_B [1:4]

USB1_PCTL0

USB1_PCTL1

USB1_SUSPEND_STP

TSEC2_TXD[6:4]

UART_SOUT[1:2]

Impact: Board connectivity test tool cannot put the pin in High Z.

Workaround: Use the HIGH Z command to put all pins of the chip in High Z.

Fix plan: Fixed in Rev 1.1

JTAG3: TRST VIH input level may glitch and cause TAP controller to reset.

Description: Devices: MPC8349E, MPC8347E, MPC8343E

The VIH input level on TRST may glitch during JTAG boundary scan testing when all I/O are switching. This may cause the TAP controller to reset.

Impact: Will cause the BSDL test to fail.

Workaround:
- For TBGA package, keep VIH higher than 2.7 V, or use an active driver with output voltage of 3.3 V.
- For PBGA package, use an active driver with output voltage of 3.3 V.

Fix plan: No plans to fix

ARB1: Bus hangs under certain transaction pattern

Description: Devices: MPC8349E, MPC8347E, MPC8343E

A logic error in the CSB arbiter may cause bus to hang when certain combination of transaction happens.

Impact: System locks up when it encounter the particular transaction pattern.

Workaround: Immediately after boot up, set HID2[10] to 1, SPCR[14] to 1. Set ACR[10:11] to 10 for CSB:CPU clock ratio of 1:1 and 1:1.5. For clock ratio 1:2 and above, ACR[10:11] needs to be set to 01.

Fix plan: Fixed in Rev 1.1.

ARB3: Changing the value of ACR[PIPE_DEP] can cause the arbiter to generate false data time out

Description: Devices: MPC8349E, MPC8347E, MPC8343E

If the pipeline depth is changed to any value other than zero by writing to ACR[PIPE_DEP], and the next transaction is acknowledged or retried at the same cycle that the actual pipe depth value change is registered in the arbiter logic, the arbiter internal logic will be broken.

Impact: The arbiter generates a false data time out.

Workaround: If the system is configured by the e300 CPU:

- Make sure no other masters are active when pipe depth is configured.
- Add a loop of 100 NOP instructions following the store to ACR. In order to avoid an instruction fetch after the ACR write, make sure the instruction cache is on and that the ACR write and the loop of 100 NOPs are in the same cache line.This will ensure that no instruction is fetched after the ACR write and that, therefore, CSB is idle while the pipe depth is being updated.

For PCI agent mode, use an external PCI host to configure the system. Hold the e300 in core disable mode by setting the CORE_DIS bit in RCWH. Set ACR[PIPE_DEP] to the desired value and then wait for at least 1 µs before issuing further transactions to the device. At this point the e300 core can be enabled to fetch its boot code by clearing ACR[CORE_DIS].

If possible, use the I^2C boot sequencer to configure initial settings of the system, including ACR[PIPE_DEP].

Fix plan: No plans to fix

IPIC1: Feature that the highest priority interrupt and two highest priority interrupts from each group can be programmed to support $\overline{cint}$ and $\overline{smi}$ does not work

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Due to a logic error, highest priority interrupt and two highest priority interrupts from each group cannot be programmed to support $\overline{cint}$ and $\overline{smi}$.

Impact: System cannot support $\overline{cint}$ and $\overline{smi}$ feature.

Workaround: Do not use this feature. Always program SICFR[HPIT], SECNR[MIXA0T], SECNR[MIXA1T], SECNR[MIXB0T], SECNR[MIXB1T], SICNR[SYSA0T], SICNR[SYSA1T], SICNR[SYSB0T], SICNR[SYSB1T], SICNR[SYSC0T], SICNR[SYSC1T], SICNR[SYSD0T], SICNR[SYSD1T] to 00.

Fix plan: Fixed in Rev 3.x

DMA1: DMA writes to prefetchable buffer in IOS might cause memory coherence problem

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When the DMA writes to an address that matches a prefetched buffer in the IOS, this buffer should be invalidated for coherence. However, the buffer is not invalidated, and the out-of-date contents of the buffer could be read by the PCI port at a later time.

Impact: System has a memory coherence problem if not following the workaround.

Workaround: Do not use the DMA to write to an area of memory that will be read by an external PCI master and is marked as prefetchable.

Fix plan: Fixed in Rev 1.1

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

Description: Devices: MPC8349E, MPC8347E, MPC8343E

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

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

Examples of error condition are as follows:

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

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

Workaround: Use one of the following options:

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

Fix plan: No plans to fix

RESET1: $\overline{\text{HRESET}}$ is not synchronized prior to use in the internal state machines

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When the $\overline{\text{HRESET}}$ pin is driven from an external source such as COP, it is not sampled before being used in the internal state machines. As a result, if $\overline{\text{HRESET}}$ negation happens at the same time as the rising edge of input clock (PCI_SYNC_IN), there could be a case in which some logics see $\overline{\text{HRESET}}$ as being asserted and others see it as being negated. If this is the case, the chip will be stuck at reset.

Impact: Chip could get stuck at reset if external $\overline{\text{HRESET}}$ negation is not synchronized to PCI_SYNC_IN.

Workaround:

- Synchronize the external $\overline{\text{HRESET}}$ to PCI_SYNC_IN. Replace the pull up with a strong (300 Ω) pull-up to minimize the signal rise time and meet setup time to next rising edge of PCI_SYNC_IN (minimum period is 15 ns).

-or-

- Shorten the external $\overline{\text{HRESET}}$ assertion to be > 32 PCI_SYNC_IN clock cycles and < 100 PCI_SYNC_IN clock cycles. This way, when the external source stops driving the $\overline{\text{HRESET}}$, the chip continues to drive the $\overline{\text{HRESET}}$ and then negate synchronously with the clock. Replace the pull-up with a strong (300 Ω) pull-up to minimize the signal rise time and meet setup time to next rising edge of PCI_SYNC_IN (minimum period is 15 ns).

The benefit of work around 2 is that it does not need to sync with PCI_SYNC_IN.

Fix plan: Fixed in Rev 1.1

RESET2: Change of LBIUCM or DDRCM bit value in reset configuration word will cause the device not able to recover from $\overline{\text{HRESET}}$

Description: Devices: MPC8349E, MPC8347E, MPC8343E

If RCWL[LBIUCM] or RCWL[DDRCM] bit value is changed in the RCW source, and then HRESET is applied to the device, the device will try to load the new RCW values. In this case the new value for LBIUCM or DDRCM will be different than the value that was loaded during PORESET sequence. This will cause the device to not be able to transfer data on the DDR or Local bus interface (the one that had been changed).

Impact: Users cannot change LBIUCM and DDRCM during $\overline{\text{HRESET}}$

Workaround: Do not change LBIUCM and DDRCM. If a change is needed, apply $\overline{\text{PORESET}}$ afterwards

Fix plan: Fixed in Rev 3.x

RESET3: External Soft reset functionality is not functional

Description: Devices: MPC8349E, MPC8347E, MPC8343E

External Soft reset is defined such that the DDR controller and the local bus controller will not be reset in the event of an external soft reset. However, the internal bus masters and the internal bus components will be held in reset immediately and as long as external soft reset event is valid and the soft reset sequence is in progress. As a result, if the soft reset event happens in the middle of an outstanding write transaction which is targeted to the DDR main memory or to one of the local bus slaves, the internal bus component will transition to reset state while it needs to supply the data to be written and as a result wrong data could be written to the main memory or to the local bus slave.

External Soft reset is defined such that it has no effect on system configuration registers, including IMMRBAR. As a result, software should behave different in regards to IMMRBAR initialization when recovering from soft reset as opposed to recovering from hard reset (in which the IMMRBAR is guaranteed to be located in its default mapping). Since software cannot tell if it is recovering from soft reset or hard reset without reading the platform's RSR register, and since software has to know the IMMRBAR value in order to read the RSR register, there is no way to recover from soft reset (unless the IMMRBAR is kept in its default value at all time).

Impact: Recovery from a soft reset event is not guaranteed.

Workaround: Do not use soft reset at all. Use hard reset or power on reset. SRESET signal must be used as an output-only signal. Software must not write 1 to the RCR[SWSR] bit.

Fix plan: No plans to fix

PMC1: Entering low power mode may result in loss of data in DDR memory

Description: Devices: MPC8349E, MPC8347E, MPC8343E

When entering low power mode and DDR self-refresh is enabled in DDR controller, the chip could hang in a state where the MCK clocks are off but the self-refresh command is not provided to DDR memories. This state can happen only if one of the internal CSB masters requests the bus within a short period after the CPU initiated a low power mode request.

Impact: Data in DDR memory can be lost.

Workaround: Disable all internal masters prior to initiating low power request.

Fix plan: Fixed in Rev 1.1.

THERM1: THERM0 pin does not work

Description: Devices: MPC8349E, MPC8347E, MPC8343E

THERM0 pin is used to measure the junction temperature. This pin does not work.

Impact: Junction temperature cannot be measured with THERM0 pin.

Workaround: None

Fix plan: Fixed in Rev 1.1

GPIO1: Certain GPIO pins are not functioning if TSEC1 is using RGMII and RTBI mode

Description: Devices: MPC8349E, MPC8347E, MPC8343E

Some GPIO1 pins are multiplexed with TSEC2 pins. The power supply of those pins is mistakenly tied to the TSEC1 pins power supply. So if TSEC1 is configured in RGMII or RTBI mode, then its power supply is 2.5 V. As a result, those GPIO pins have a 2.5 V power supply and cannot function as GPIO which requires a 3.3 V power supply. Pins affected are GPIO1[12:20] and GPIO1[22:23].

Impact: When TSEC1 uses RGMII or RTBI mode, some GPIO1 pins do not function. For more information, see TSEC22.

Workaround: No.

Fix plan: Fixed in Rev 3.x

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

Document Number: MPC8349ECE
Rev. 19
08/2011

freescale semiconductor