Mask Set Errata for Mask REVA

Introduction

This report applies to mask REVA for these products:

- MPC5553

ID before 15 MAY 2008	ID from 15 May 2008 to 30 JUNE 2010	ID after 1 JULY 2010	Errata Title
645	6448	1694	BAM: Peripheral Bridge A not initialized as guarded
N/A	12488	2279	BAM: Pull RXD_A high during CAN serial boot mode
2297	7062	1722	BAM: Serial download unavailable to last 16 bytes (4 words) of System RAM
2237	7461	2114	DMA: Dynamic writes to DMA control register can induce preemption failure
1123	6934	575	DSPI: Changing CTARs between frames in continuous PCS mode causes error
4031	5922	621	DSPI: DSPI D PCS[3:4] are slow speed pins
4022	11100	1154	DSPI: DSPI_B pins split to separate supply, VDDEH10
N/A	10483	1103	DSPI: PCS Continuous Selection Format limitation
2264	11097	1147	DSPI: Using DSPI in DSI mode with MTO may cause data corruption
N/A	9739	1082	DSPI: set up enough ASC time when MTFE=1 and CPHA=1
2379	7097	1756	EBI: Calibration pads are 1 ns slower than EBI
2823	7049	1708	EBI: Do not access external resources when the EBI is disabled
3111	11099	1151	EBI: Dual controller mode cannot be guaranteed under all conditions
1119	6826	1360	EBI: Incorrect write data on transaction following a burst access with error.
3839	5762	1844	EBI: Timed out accesses (external TA only) may generate spurious TS_B pulse
1651	7428	2773	ECSM: ECC error reported on prefetches outside the flash
2235	7447	2089	ECSM: ECC event can get reported incorrectly

Table continues on the next page...

ID before 15 MAY 2008	ID from 15 May 2008 to 30 JUNE 2010	ID after 1 JULY 2010	Errata Title
2236	7429	2056	ECSM: ECC event can report incorrect address
3819	5715	1741	EQADC : 25% calibration channel sampling requires at least 64 sampling cycles
1742	6968	652	EQADC: 50% reference channels reads 20 mv low
1921	7420	2046	FBIU: Disable prefetch before invalidating the flash BIU buffers
		2455	FEC: Fast Ethernet Controller (FEC) start-up issue
2049	7440	2077	FEC: Back to back reads of the same buffer descriptor are not coherent and may cause unexpected results
N/A	19475	1281	FEC: Duplicate Frame Transmission
746	6588	1129	FEC: Late collision, retry limit, and underrun interrupts will not trigger on consecutive transmit frames
2544	6891	1601	FEC: do not access the module address space in the 208 or 324 packages
741	6519	2340	FEC: slot time is designed for 516 bit times; deviation from the 802.3
1920	7413	989	FLASH: Disable Prefetch during programming and erase
2371	7439	2075	FLASH: Large blocks limited to 1,000 Program/erase cycles
2419	6946	605	FLASH: Minimum Programming Frequency is 25 MHz
1745	7444	2083	FLASH: The ADR register may get loaded with a flash address even through no ECC error has occurred
715	6764	1111	FMPLL: LOLF can be set on MFD change
N/A	8014	2181	FMPLL: Non-zero pre-divider values can cause PLL lock failure
N/A	18087	1232	FMPLL: Reset may not be negated if an external reset occurs during a software initiated PLL relock sequence
		6531	FMPLL: Selecting GPIO mode on RSTCFG/PLLCFG[0:1] may cause PLL failure after a reset
		3407	FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1
4414	5952	1557	FlexCAN: Corrupt ID may be sent in early-SOF condition
N/A	23304	2685	FlexCAN: Module Disable Mode functionality not described correctly
1428	6921	551	FlexCAN: Transmit Buffers May Freeze / missing frame
1831	6981	683	FlexCAN: receive time stamp may be incorrect
N/A	34749	2424	FlexCAN: switching CAN protocol interface (CPI) to system clock has very small chance of causing the CPI to enter an indeterminate state
1037	6852	1488	FlexCAN: writing to an active receive MB may corrupt MB contents
5244	6276	817	JTAGC: EVTI and RDY require TCK to toggle
331	6422	1666	JTAGC: SAMPLE instruction does not sample input data during board boundary scan testing.
2526	6969	653	MPC5553: SIU_MIDR Revision field is 0x0010, NPC_DID[PIN]=0x53
N/A	40092	6726	NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled
1800	7216	1957	NPC: MCKO_DIV can be set to 0x0 (1X MCKO)

Table continues on the next page...

ID before 15 MAY 2008	ID from 15 May 2008 to 30 JUNE 2010	ID after 1 JULY 2010	Errata Title
2273	7451	2096	NZ6C3: No sync message generated after 255 direct branch messages in history mode
1362	7430	2058	NZ6C3: Branch Trace History field on PCM message is zero
2706	7456	2103	NZ6C3: Data Trace of stmw instructions may cause overruns
1707	7410	985	NZ6C3: Incorrect data traced on misaligned little endian store
410	7396	950	NZ6C3: Nexus stall mode may not prevent all Nexus overflow conditions
108	6899	1618	NZ6C3: No indication of an exception causing a Nexus Program Trace (PT) message as opposed to a retired branch instruction causing a PT message.
2097	7424	2052	NZ6C3: RFM not sent for history buffer overflow caused by 'evsel'
4354	11102	1159	Pad Ring: ESD specifications are not met
4381	5934	1528	Pad Ring: Pin behavior during power sequencing
63	6700	488	Pad Ring: Possible poor system clock just after POR negation.
64	7224	1969	Pad Ring: RSTOUT is 3-stated during the power-on sequence.
		3377	Pad Ring:Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge
N/A	8913	1031	e200z6: Cache Invalidate/Clear Aborts due to masked core Interrupt
507	7419	2044	e200z6: Core renamed from e500z6
1323	11095	1143	e200z6: Data Storage Exception taken instead of Machine Check
3413	7350	846	e200z6: Debug interrupt (IVOR15) can be taken erroneously
1232	6871	1536	e200z6: JTAG Part Identification is 0x2
674	7380	914	e200z6: MFSPR may read prior value of SPEFSCR
2312	7068	1729	e200z6: MMU has 32 Table Entries
3474	7352	851	e200z6: wrong exception taken after FPU instruction preceding a lmw
		3421	e200z: Cache returns value, even when disabled
5093	11091	1136	eDMA: BWC setting may be ignored between 1st and 2nd transfers and after the last write of each minor loop.
1332	7417	2043	eDMA: Unexpected start of channel 15 during a spurious preemption
1970	6778	1168	eMIOS: Asynchronous reads of the A and B registers in IPM and IPWM modes may not be coherent
2013	7083	1747	eMIOS: Comparators A and B enabled by writing to A2 and B2 in DAOC mode
2305	7147	1834	eMIOS: OPMWC unable to produce close to 100% duty cycle signal
1278	6788	1201	eMIOS: Possible incoherent access in PEA mode
860	6655	421	eMIOS: Possible incoherent accesses in IPWM/IPM modes
2251	6992	702	eMIOS: Writes to EMIOS_MCR register in IPM and IPWM modes may cause incoherent reads
2878	7114	1773	eQADC: conversions of muxed digital/analog channels close to the rail
N/A	8632	1013	eSCI : Automatic reset of the LIN state machine cause incorrect transmission

Table continues on the next page...

ID before 15 MAY 2008	ID from 15 May 2008 to 30 JUNE 2010	ID after 1 JULY 2010	Errata Title
N/A	27326	1381	eSCI: LIN Wakeup flag set after aborted LIN frame transmission
N/A	17048	1221	eSCI: LIN bit error indicated at start of transmission after LIN reset
N/A	8635	1017	eSCI: LIN fast bit error detection causes incorrect LIN reception
N/A	8173	2190	eSCI: LIN slave timeout flag STO not asserted if CRC is received too late
7064	4968	1614	eSCI: Low pulse on LIN RX line may prevent assertion of transmit data ready flag ESCI_SR[TXRDY]
1846	6903	1634	eTPU: LAST can fail on consecutive teeth
2477	6965	644	eTPU: MISSCNT can fail on sequential physical teeth
7331	3377	112	eTPU: Prescaler phase shift between TCR1 and TCR2
3150	5710	1728	eTPU: STAC bus export may skip 1 count
1807	6634	380	eTPU: TCR2, LAST can negate early in High Rate mode

e1694: BAM: Peripheral Bridge A not initialized as guarded

Errata type: Information

Description: The Memory Management Unit (MMU) region for Peripheral Bridge A is initialized by the Boot Assist Module (BAM) as not guarded. Some peripherals, such as the eMIOS, have registers which have read side effects. While the e200z6 does not issue speculative reads, if a future processor core in the MPC5500 family issued reads speculatively, then non-coherent data can be read from the peripheral. While the BAM does not access Peripheral Bridge A, the MMU entries that it configures are meant to work for all MPC5500 family members.

Workaround: For future compatibility, configure the MMU entry for Peripheral Bridge A to be guarded.

e2279: BAM: Pull RXD_A high during CAN serial boot mode

Errata type: Errata

Description: The Boot Assist Module (BAM) disables the internal pull up resistor on serial port A receive data pin (eSCI RXD_A) during serial boot mode operation. If the pin is not actively driven by system, its input level may drift below low threshold and be recognized as a valid eSCI boot operation thus preventing CAN boot mode selection.

Workaround: Always drive the eSCI RXD_A pin high during CAN serial boot mode. A external 10K pullup resistor can be used for this purpose.

e1722: BAM: Serial download unavailable to last 16 bytes (4 words) of System RAM

Errata type: Information

Description: When using the BAM Serial boot download feature, the BAM initializes an additional 4 32-bit words after the end of the downloaded records. This is done to insure that if the core fetches the last instruction of the downloaded code from the internal SRAM while executing the code, it will not prefetch instructions from memory locations that have not been initialized.

Note: if the download image has the exact same size as the internal SRAM, the 20 bytes at the beginning of the SRAM will be written with zero value due to incomplete memory decoding.

Workaround: When using the Serial download feature of the BAM, make sure that the maximum address of the downloaded code does not exceed the end address of the SRAM minus 16 bytes or the last address of the Memory Management Unit (MMU) entry minus 16 bytes (for devices with MMU and the SRAM MMU setting less than the full SRAM size), whichever is smaller.

e2114: DMA: Dynamic writes to DMA control register can induce preemption failure

Errata type: Errata

Description: If the DMA control register (EDMA_CR) is written while a channel is in the process of being preempted by a higher priority channel, the preemption process may be treated as spurious. In this case, the original channel is not preempted but its priority and preemption enable bit are temporarily replaced with those of the channel that caused the spurious preemption. After the lower priority channel completes its transfer, its original priority is restored and the higher priorty channel starts its transfer.

This temporary priority change may cause further blocking of higher priority preempting channels.

Workaround: Do not use the channel preemption feature or if you use preemption, don't write the DMA control register when a preemptable channel is executing.

e575: DSPI: Changing CTARs between frames in continuous PCS mode causes error

Errata type: Errata

Description: Erroneous data could be transmitted if multiple Clock and Transfer Attribute Registers (CTAR) are used while using the Continuous Peripheral Chip Select mode (DSPIx_PUSHR[CONT=1]). The conditions that can generate an error are:

1) If DSPIx_CTARn[CPHA]=1 and DSPIx_MCR[CONT_SCKE = 0] and DSPIx_CTARn[CPOL, CPHA, PCSSCK or PBR] change between frames.

2) If DSPIx_CTARn[CPHA]=0 or DSPIx_MCR[CONT_SCKE = 1] and any bit field of DSPIx_CTARn changes between frames except DSPIx_CTARn[PBR].

Workaround: When generating DSPI bit frames in continuous PCS mode, adhere to the aforementioned conditions when changing DSPIx_CTARn bit fields between frames.

e621: DSPI: DSPI D PCS[3:4] are slow speed pins

Errata type: Information

Description: The eMIOS[10:11]/PCSD[3:4]/GPIO[189:190] pins have a pad type of SH (slow speed pad) instead of MH (medium speed pad). While the eMIOS function normally does not require a medium speed pad, when the pin is configured as the Deserial Serial Peripheral Interface D Peripheral Chip Select (DSPI_PCSxD), the slow pad may limit the maximum speed of the DSPI port.

Workaround: Either don't use the DSPI_D PCS functions on these pins or limit the frequency of the DSPI port to account for the difference in slew rate of the pins. The slow pads have a slew rate of 15 to 200 ns and the medium speed pads have a slew rate of 8 to 100 ns (both with a 50 pF load) depending on the setting of the Slew Rate Control bits in the Pad Configuration register (PCRx[SRC]).

e1154: DSPI: DSPI_B pins split to separate supply, VDDEH10

Errata type: Information

Description: The DSPI_B SINB, SOUTB, SCKB, PCS_B[0:2] were separated from the VDDEH6 and are now powered by the new power supply pin VDDEH10. Ball J23 on the 416 package was changed from being a duplicate VDDEH6 pin to being a separate VDDEH10 supply pin. 324 pin package drawings show the VDDE10 ball placement. VDDEH6 and VDDEH10 are combined/shorted internally on 208 packages.

Workaround: For compatibility to the MPC5554, always power VDDEH6 and VDDEH10 from the same power supply (3.0 to 5.25 volts). If compatibility is not required to the MPC5554, VDDEH10 and VDDEH6 can be supplied by different voltage supplies. This allows one DSPI to operate at a different voltage than the other DSPI modules (3.3 and 5 volts, for example).

e1103: DSPI: PCS Continuous Selection Format limitation

Errata type: Errata

Description: When the DSPI module has more than one entry in the TX FIFO and only one entry is written and that entry has the CONT bit set, and continuous SCK clock selected the PCS levels may change between transfer complete and write of the next data to the DSPI_PUSHR register.

For example:

If the CONT bit is set with the first PUSHR write, the PCS de-asserts after the transfer because the configuration data for the next frame has already been fetched from the next (empty) fifo entry. This behavior continues till the buffer is filled once and all CONT bits are one.

Workaround: To insure PCS stability during data transmission in Continious Selection Format and Continious SCK clock enabled make sure that the data with reset CONT bit is written to DSPI_PUSHR register before previous data sub-frame (with CONT bit set) transfer is over.

e1147: DSPI: Using DSPI in DSI mode with MTO may cause data corruption

Errata type: Errata

Description: Using the DSPI in Deserial Serial Interface (DSI) Configuration (DSPIx_MCR[DCONF]=0b01]) with multiple transfer operation (DSPIx_DSICR[MTOE=1]) enabled, may cause corruption of data transmitted out on the DSPI master if the clock Phase is set for leading edge capture DSPIx_CTARn[CPHA]=0. The first bit shifted out of the master DSPI into the slave DSPI module will be corrupted and will convert a '0' to read as a '1'.

Workaround: There are three possible workarounds for this issue.

1) Select CPHA=1 if suitable for external slave devices.

2) Set first bit to '1', or ignore first bit. This may not be a workable solution if this bit is required.

3) Connect SOUT from the master to SIN of the first slave externally instead of using internal signals. This is achieved by setting the DSPI Input Select Register (SIU_DISR) to set the SINSELx field of the first slave DSPI to '00' and configuring this slave's SIN pin and master SOUT pin as DSPI SIN/SOUT functions respectively. This workaround is suitable only if these two signals are available to be connected externally to each other.

e1082: DSPI: set up enough ASC time when MTFE=1 and CPHA=1

Errata type: Information

Description: When the DSPI is being used in the Modified Transfer Format mode (DSPI_MCR[MTFE]=1) with the clock phase set for Data changing on the leading edge of the clock and captured on the following edge in the DSPI Clock and Transfer Attributes Register (DSPI_CTARn[CPHA]=1), if the After SCK delay scaler (ASC) time is set to less than 1/2 SCK clock period the DSPI may not complete the transaction - the TCF flag will not be set, serial data will not received, and last transmitted bit can be truncated.

Workaround: If the Modified Transfer Format mode is required DSPI_MCR[MTFE]=1 with the clock phase set for serial data changing on the leading edge of the clock and captured on the following edge in the SCK clock (Transfer Attributes Register (DSPI_CTARn[CPHA]=1) make sure that the ASC time is set to be longer than half SCK clock period.

e1756: EBI: Calibration pads are 1 ns slower than EBI

Errata type: Information

Description: The calibration bus outputs and input setup time is 1ns longer than the equivalent normal External bus signals. Therefore, the electrical specifications need to be added to the data sheets for the calibration signals.

Workaround: For synchronous (to CLKOUT) peripherals on the calibration pads, make certain that the bus will meet the new electrical specification.

e1708: EBI: Do not access external resources when the EBI is disabled

Errata type: Information

Description: When the external bus is disabled in the External Bus Interface Module Control Register (EBI_MCR[MDIS] = 1), accesses through the EBI will not terminate and the master requesting the access will not request another one.

Workaround: Do not disable the EBI or do not allow accesses to the external bus through Memory Management Unit (MMU) settings in the core. Other internal bus masters (such as DMA) bypasses the MMU and therefore these accesses will hang the external bus if the destination is in the external bus address map.

e1151: EBI: Dual controller mode cannot be guaranteed under all conditions

Errata type: Errata

Description: In dual controller mode, the specification for the phase relationship between EXTAL and CLKOUT is +/- 1 ns, however this does not allow adequate set up and hold times to guarantee successful operation of the external bus to a second MCU.

Workaround: Do not use in Dual Controller mode.

e1360: EBI: Incorrect write data on transaction following a burst access with error.

Errata type: Errata

Description: If a write access that is terminated by an error (TEA) is immediately followed by a back-to-back pipelined write transaction, incorrect data (data from the terminated transaction written instead of the second write transaction) could be written to memory. This condition could occur if one of the following occurs:

1) A Cache-enabled write access to a chip-selected (usually external memory) or non-chip-selected region (usually external slave MCU), with external TEA asserted, is followed by pipelined write access.

or

2) A Cache-enabled write access to a non-chip-select region, with EBI bus monitor timeout (which generally indicates a severe system problem - memory not present or responding), followed by pipelined write access.

Workaround: Avoid situations in which a burst write can terminate with an error and immediately be followed by an additional write by not using the TEA functionality for external accesses and don't cache-enable non-chip-select external address regions.

e1844: EBI: Timed out accesses (external TA only) may generate spurious TS_B pulse

Errata type: Errata

Description: When an external Transfer Acknowledge (TA) access times out, there is a boundary case where the External Bus Interface (EBI) asserts a Transfer Start (TS) pulse as if starting another access, even if no other internal request is pending. The boundary case is when the access is part of a "small access" set (sequence of external accesses to satisfy 1 internal request), and when the external TA arrives around the same cycle (+/- 1 clkout cycle) as the bus monitor timeout (BMT).

Most EBI signals will stay negated during this erroneous transfer (CS, OE, WE, BDIP). However, along with TS assertion, RD_WR may also assert (for 1 cycle only, during this phantom TS), if the prior access that timed out was a write. This condition can generate an erroneous write transfer (with CS negated). The address (ADDR pins) will be incremented to the address of the next small access transfer that would have been performed, and the value driven by the EBI on the DATA bus (if a write) may change. Busy Busy (BB) may be asserted along with the phantom TS (if external master modes is enabled in the EBI Module configuration Register, SIU_MCR[EXTM]=1), and the Transfer Size (TSIZ) value may change.

Internally, the EBI terminates the timeout access, and the internal state machine goes to IDLE after the timeout access. So the EBI will not be "hung" after the spurious TS, and the EBI does respond properly to future internal or external requests.

However, the side effect of the spurious TS is that it may cause an external non-chip-select device to think an access is being performed to it, resulting in 1 of 2 bad effects (depending on RD_WR value during spurious TS):

1) RD_WR high (read): ext. device may drive back read data some number of cycles later, possibly conflicting with a future real access (e.g. write) that might have started by that time.

2) RD_WR low (write): ext. device may get an erroneous write performed to it

Note that the soonest possible TS for a real transfer (after the timeout transfer), is 2 cycles after the spurious TS (so 1 cycle gap), meaning this Bug will never result in a 2-cycle TS pulse.

Workaround: Do not enable bus monitor in the EBI Bus Monitor Control Register (keep SIU_BMCR[BME]=0), unless at least 1 of the following 3 conditions can be met:

1) The external TA will never be asserted from external device within 1 cycle of when the access would be timing out (see NOTE below)

2) No internal requests greater than external bus size will be performed (e.g. doing data-only fetches of 32 bits or less on 32-bit data bus or 16 bits or less on a 16 bit bus only, so a "small access" could never occur).

3) The side effect of this TS pulse driven to non-CS device is judged to be tolerable in system after a timeout error occurs; depends on spec of external device and user requirements for data coherency after a timeout error occurs.

NOTE: Of the 3 above, #1 is easiest to achieve in most systems. If the maximum possible TA latency of the external device is known, the user just needs to set the BMT period more than (external device maximum latency + 2), and this condition will not occur.

e2773: ECSM: ECC error reported on prefetches outside the flash

Errata type: Errata

Description: Accessing the last pages of the internal flash array may cause the flash controller to prefetch past the end of the array if the prefetch limits set up in the flash bus interface control register (FLASH_BIUCR) is greater than 1. This will return an ECC error from the array which will be registered in the Error Correction Module (ECSM). The core will take a data storage exception (IVOR2) for data accesses, an illegal instruction exception (IVOR3), or in other cases, it will be ignored.

Workaround: Since there are only 2 prefetch buffers in the flash, do not use the last 64 bytes of the internal flash.

e2089: ECSM: ECC event can get reported incorrectly

Errata type: Errata

Description: The Error Correction Status Module may report inaccurate attributes for a single or multi-bit error that occurs on a read after write to the internal SRAM. The attributes would be report incorrectly when the ECC event occurs during a read that results in 2 wait states (one for the previous write and then one for the read).

Workaround: The user should understand that the attributes for the ECC event may be incorrect.

e2056: ECSM: ECC event can report incorrect address

Errata type: Errata

Description: An incorrect failing address for an Error Correction event may be reported when performing a wrapped 8-bit burst read from the internal SRAM. The byte that caused the ECC event will be included in the 64-bit address reported, but may not be the exact address reported.

Workaround: The user should be aware that the address reported for an ECC event from the Error Correctionand Status Module (ECSM) may not be completely correct. For example, an ECC error that occurs on address 0x4000_0000 could be reported as 0x4000_0004, if the wrapped burst began on address 0x4000_0004, but received the ECC error when the burst wrapped back to address 0x4000_0000.

e1741: EQADC : 25% calibration channel sampling requires at least 64 sampling cycles

Errata type: Information

Description: The 25%*(VRH-VRL) calibration channel (ADC channel 44) will not convert to specification with an ADC sample time less than 64 cycles.

Workaround: For accurate calibration, the 25% calibration channel should be converted using the Long Sample Time (LST) setting for either 64 or 128 ADC sample cycles in the ADC Conversion Command Message (LST = 0b10 or 0b11).

e652: EQADC: 50% reference channels reads 20 mv low

Errata type: Information

Description: The equation given for the definition of the 50% reference channel (channel 42) of the Enhanced Queued Analog to Digital Converter (eQADC) is not correct. The 50% reference point will actually return approximately 20mV (after calibration) lower than the expected 50% of difference between the High Reference Voltage (VRH) and the Low Reference Voltage (VRL).

Workaround: Do not use the 50% point to calibrate the ADC. Only use the 25% and 75% points for calibration.

After calibration, software should expect that the 50% Reference will read 20 mV low.

e2046: FBIU: Disable prefetch before invalidating the flash BIU buffers

Errata type: Errata

Description: Flash programming or erasing will automatically invalidate the flash prefetch buffers in the Flash Bus Interface. When prefetching is enabled while the buffers are being invalidated, a prefetch could be initiated that could result in a prefetch data acknowledge for a subsequent transfer and result in incorrect data being returned on that subsequent transfer.

Workaround: Disable prefetching by clearing either the master prefetch enable for all masters in the FBIU Control Register (FLASH_BIUCR[MnPFE]=0b00000), clearing both the Data Prefetch Enable and the Instruction Prefetch Enable (FLASH_BIUCR[DPEN] AND FLASH_BIUCR[IPFEN]=0b00), or setting the prefetch limit to zero (FLASH_BIUCR[PFLIM]=0b000) before performing any program or erase operation to the flash. This means that prefetching should be disabled before starting a program or erase operation, or when turning the buffers off and then on again.

e2455: FEC: Fast Ethernet Controller (FEC) start-up issue

Errata type: Errata

Description: The Fast Ethernet Controller (FEC) module may be in a non-functional state after device power-on. In this state, writes to FEC registers will not complete and no exceptions will be raised. The FEC module is the only module on the device affected by this issue.

Workaround: During execution of boot code, perform a write to an FEC register. Verify that the value of the register has been written as expected. If the write was executed successfully, the FEC module is functional and no further action is needed. If the write was not executed successfully, cycle all power supplies using external circuitry.

e2077: FEC: Back to back reads of the same buffer descriptor are not coherent and may cause unexpected results

Errata type: Errata

Description: When consecutive reads to the same address occur in the Fast Ethernet Controller (FEC), the FEC will just return data from its read line buffer and will not cause the data to be reread from memory. This can cause a buffer descriptor failure. For example, the FEC reads a transmit buffer descriptor into the line buffer with a zero Ready (R) bit causing the transmitter to become idle. Software then prepares that buffer, sets the R bit in the buffer descriptor, and writes the Transmit Descriptor Active Register (FEC_TDAR) to begin transmission. The FEC reads the transmit buffer descriptor from the FEC read line buffer instead of memory and therefore does not see the R bit that was set by software. The problem also exists with the receiver and the Empty (E) bit of the receive buffer descriptor.

Workaround: Software should prepare an alternate transmit buffer descriptor with a zero Ready (R) bit and an alternate receive buffer descriptor with a zero Empty (E) bit.

When the transmitter goes idle (FEC_TDAR[X_DES_ACTIVE] = 0) due to the FEC encountering a transmit buffer descriptor with a zero R bit, use the following procedure to resume transmission:

1. Prepare transmit buffers and transmit buffer descriptors as needed.

2. Read and save the value from XDES_ADDR (FEC_BASE + 0x1C8 - internal pointer to the next transmit buffer descriptor to be opened).

3. Write the address of the alternate transmit buffer descriptor to both XDES_ADDR and XDES_CLOSEP_ADDR (FEC_BASE + 0x18C - internal pointer to the next transmit buffer descriptor to be closed).

4. Activate the transmitter by writing to FEC_TDAR. This will cause the FEC to load the alternate buffer descriptor.

5. Wait for the transmitter to go idle (FEC_TDAR[X_DES_ACTIVE] = 0) due to the zero R bit.

6. Write the saved value read in step 2 to XDES_ADDR and XDES_CLOSEP_ADDR.

7. Activate the transmitter by writing to FEC_TDAR.

When the receiver goes idle (FEC_RDAR[R_DES_ACTIVE] = 0) due to the FEC encountering a receive buffer descriptor with a zero E bit, use the following procedure to resume reception:

1. Prepare receive buffers and receive buffer descriptors as needed.

2. Read and save the value from RDES_ADDR (FEC_BASE + 0x1C4 - internal pointer to the next receive buffer descriptor to be opened).

3. Write the address of the alternate receive buffer descriptor to both RDES_ADDR and RDES_CLOSEP_ADDR (FEC_BASE + 0x190 - internal pointer to the next receive buffer descriptor to be closed).

4. Activate the receiver by writing to FEC_RDAR. This will cause the FEC to load the alternate buffer descriptor.

5. Wait for the receiver to go idle (FEC_RDAR[R_DES_ACTIVE] = 0) due to the zero E bit.

6. Write the saved value read in step 2 to RDES_ADDR and RDES_CLOSEP_ADDR.

7. Activate the receiver by writing to FEC_RDAR.

e1281: FEC: Duplicate Frame Transmission

Errata type: Errata

Description: In some cases, the Fast Ethernet Controller (FEC) will transmit single frames more than once. The FEC fetches the transmit buffer descriptors (TxBDs) and the corresponding Tx data continuously until the Tx FIFO is full. It does not determine whether the TxBD to be fetched is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. To remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. The FEC may fetch a BD from memory that has already been processed but not yet written back (it is read a second time with the R bit still set). In this case, the data is fetched and transmitted again.

Workaround: Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for large or small frames, one of the following must be true:

* The FEC software driver ensures that the Ready bit cleared in at least one TxBD.

* Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched.

* The FEC software driver ensures a minimum frame size, n. The minimum number of TxBDs is then rounded up to the nearest integer (although the result cannot be less than 3). The default Tx

FIFO size is 192 Bytes; this size is programmable.

e1129: FEC: Late collision, retry limit, and underrun interrupts will not trigger on consecutive transmit frames

Errata type: Errata

Description: The late collision (LC), retry limit (RL), and underrun (UN) interrupts of the Fast EtherNet Controller (FEC) will not trigger on consecutive transmit frames. For example, if back-to-back frames cause a transmit underrun, only the first frame will generate an underrun interrupt. No other underrun interrupts will be generated until a frame is transmitted that does not underrun or the FEC is reset.

Workaround: Since late collision, retry limit, and underrun errors are not directly correlated to a specific transmit frame, in most cases a workaround for this problem is not needed. If a workaround is required, then there are two independent workarounds:

- Ensure that a correct frame is transmitted after a late collision, retry limit, or underrun errors are detected.

- Perform a soft reset of the FEC by setting ECR[RESET] when a late collision, retry limit, or underrun errors are detected.

e1601: FEC: do not access the module address space in the 208 or 324 packages

Errata type: Errata

Description: Accesses to the Fast Ethernet Controller (FEC) module address space in packages that do not include the FEC module pins, may not terminate the cycle correctly (either by a normal cycle termination or transfer error).

Workaround: Do not access the FEC memory space in 208 or 324 packages and/or set an MMU table entry to prevent the CPU from accessing this address space.

e2340: FEC: slot time is designed for 516 bit times; deviation from the 802.3

Errata type: Information

Description: The Fast Ethernet Controller (FEC) slot time is 516 bit times which is longer than the 512 bit times specified by the IEEE 802.3 standard.

If a collision occurs after the standard 512 bit times (but prior to 516 bit times), the FEC may generate a retry that a remote ethernet device may identify as late. In addition, the slot time is used as an input to the backoff timer, therefore the FEC retry timing could be longer than expected.

Workaround: No software workaround is needed or available.

e989: FLASH: Disable Prefetch during programming and erase

Errata type: Errata

Description: If a prefetch read completes in the flash bus interface (Flash_BIU) during the same cycle that a flash write is initiated, the write will not be performed.

Workaround: Disable prefetching by clearing either the master prefetch enable for all masters in the FBIU Control Register (FLASH_BIUCR[MnPFE]=0b00000), clearing both the Data Prefetch Enable and the Instruction Prefetch Enable (FLASH_BIUCR[DPEN] AND FLASH_BIUCR[IPFEN]=0b00), or setting the prefetch limit to zero (FLASH_BIUCR[PFLIM]=0b000) before performing any program or erase operation to the flash. This will ensure that all writes to the flash are done while prefetching is disabled.

e2075: FLASH: Large blocks limited to 1,000 Program/erase cycles

Errata type: Errata

Description: The electrical specification for Program/Erase cycling on large Flash blocks (all 128K blocks - Middle Address Space [MAS] blocks M0 and M1, plus High Address Space [HAS] blocks H0 to H3/H7/H11/H19 [depending on total flash size]) has been changed to 1,000 PE cycles minimum. The small blocks (16K, 48K, and 64K - Low Address Space [LAS] blocks L0-L5) are still specified as 100,000 PE cycles minimum.

The data retention specification all blocks is still 20 years for blocks cycled less than 1000 times and 5 years for blocks cycled 1001 to 100,000 cycles (1,000 for large blocks).

Workaround: Only use the small blocks for EEPROM emulation (LAS L0-L5). Do not use blocks MAS M0/M1 or HAS H0 to H3/H7/H11/H19 (depending on total flash size) for EEPROM emulation requiring greater than 1,000 Program/Erase cycles. Refer to the latest device electrical specifications (Data Sheet) dated July 2007 or later.

e605: FLASH: Minimum Programming Frequency is 25 MHz

Errata type: Information

Description: Programming and erase operations of the internal flash could fail if the clock to the flash (usually the system clock) is less than 25 MHz.

Workaround: Do not program or erase the flash when the system operating frequency is below 25Mhz.

e2083: FLASH: The ADR register may get loaded with a flash address even through no ECC error has occurred

Errata type: Information

Description: The Flash Address Register (FLASH_AR) may be loaded with a flash address when no Error Correction Code (ECC) has occurred. When an ECC does occur, the FLASH_AR is properly set.

Workaround: Check the Flash Module Control Register ECC Event Error (FLASH_MCR[EER]=1) to check for an ECC error before examining the ADR register. If an error has occurred then the ADR register data is valid. If an error has not occurred then the FLASH_AR data could change on any flash access.

e1111: FMPLL: LOLF can be set on MFD change

Errata type: Errata

Description: Normally, the Loss of Lock Flag (FMPLL_SYNCR[LOLF]) would not be set if the loss of lock occurred due to changing of the Multiplication Factor Divider bits or PREDIV bits (FMPLL_SYNCR[MFD] or [PREDIV]) or enabling of Frequency Modulation (FMPLL_SYNCR[Depth]>0b00). However, if LOLF has been set previously (due to an unexpected loss of lock condition) and then cleared (by writing a 1), a change of the MFD, PREDIV or DEPTH fields can cause the LOLF to be set again which can trigger an interrupt request if LOLIRQ bit is set.

In addition, changing the RATE bit will also set the LOLF regardless of previous conditions.

Workaround: The Loss of Lock Interrupt Request enable in the Synthesizer Control Register (FMPLL_SYNCR[LOLIRQ]) should be cleared before any change to the multiplication factor (MFD), PREDIV, modulation depth (DEPTH), or modulation rate (RATE) to avoid unintentional interrupt requests. After the PLL has locked (LOCK=1), LOLF should be cleared (by writing a 1) and LOLIRQ may be set again if required.

e2181: FMPLL: Non-zero pre-divider values can cause PLL lock failure

Errata type: Errata

Description: When configuring the MPC55xx Frequency Modulated Phase Lock Loop (FMPLL) in crystal or external reference clock mode, the system clock frequency (SYSCLK) is determined by the values programmed into the Pre-Divider (PREDIV), Multiplication Factor Divider (MFD), and Reduced Frequency Divider (RFD) fields in the FMPLL Synthesizer Control Register (FMPLL_SYNCR). If the pre-divider is set to divide by 2, 3, 4, or 5 (FMPLL_SYNCR[PREDIV] = 1, 2, 3, or 4), a condition may occur in which the FMPLL will fail to lock. Odd predividers may result in the PLL stuck in a lock routine where it can not escape. Even predividers may result in the PLL VCO frequency not being able to reach target frequencies below 110MHz. To clear this condition when it occurs, the part must be powered down.

Workaround: If a pre-divider of 2, 3, 4, or 5 must be used in order to achieve the desired system clock frequency, any write that causes a relock of the FMPLL (changing either the PREDIV or MFD) with a FMPLL_SYNCR[PREDIV] = 1, 2, 3, or 4 must occur with the current system frequency set to one-half of the crystal frequency or less through setting of the PLL RFD prior to writing the MFD or PREDIV.

NOTE: When programming the FMPLL, care must also be taken not to violate the maximum system clock frequency of the device, or the maximum and minimum frequency specifications of the FMPLL.

e1232: FMPLL: Reset may not be negated if an external reset occurs during a software initiated PLL relock sequence

Errata type: Errata

Description: During a software initiated change in frequency of the Phase Lock Loop (PLL) to a frequency greater than 112 MHz, if reset is externally asserted, it is possible that reset will never be exited.

Workaround: Do not allow external resets to be applied during a software initiated PLL frequency change that requires a relock of the PLL. A possible way to prevent this from occurring is to use a GPIO signal to gate (hold off) the RESET input to the device from the external power supply. The following sequence would have to be used:

1) drive GPIO to hold RESET negated,

2) Start the PLL lock sequence,

3) wait for PLL lock sequence to complete,

4)Toggle the GPIO pin to allow the external reset to assert the RESET pin.

e6531: FMPLL: Selecting GPIO mode on RSTCFG/PLLCFG[0:1] may cause PLL failure after a reset

Errata type: Errata

Description: If the General Purpose Input/Output (GPIO) mode is selected in software for the PLLCFG[0:1] and/or the RSTCFG pins AND the default clock reference mode (crystal reference) is not used, any RESET assertion of the device (internal or external) may corrupt the operating mode of the PLL and the microcontroller may not exit reset (RSTOUT will not negate).

Workaround: Do not enable the GPIO mode in software for the PLLCFG[0:1]/RSTCFG pins when over-riding the default (crystal) PLL clock mode (RSTCFG=0 AND PLLCFG[0:1] != 0b10).

e3407: FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1

Errata type: Errata

Description: FlexCAN does not transmit an expected message when the same node detects an incoming Remote Request message asking for any remote answer.

The issue happens when two specific conditions occur:

1) The Message Buffer (MB) configured for remote answer (with code "a") is the last MB. The last MB is specified by Maximum MB field in the Module Configuration Register (MCR[MAXMB]).

2) The incoming Remote Request message does not match its ID against the last MB ID.

While an incoming Remote Request message is being received, the FlexCAN also scans the transmit (Tx) MBs to select the one with the higher priority for the next bus arbitration. It is expected that by the Intermission field it ends up with a selected candidate (winner). The coincidence of conditions (1) and (2) above creates an internal corner case that cancels the Tx winner and therefore no message will be selected for transmission in the next frame. This gives the appearance that the FlexCAN transmitter is stalled or "stops transmitting".

The problem can be detectable only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is NO ISSUE if any of the conditions below holds:

a) The incoming message matches the remote answer MB with code "a".

b) The MB configured as remote answer with code "a" is not the last one.

c) Any MB (despite of being Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.

d) A new incoming message sent by any external node starts just after the Intermission field.

Workaround: Do not configure the last MB as a Remote Answer (with code "a").

e1557: FlexCAN: Corrupt ID may be sent in early-SOF condition

Errata type: Errata

Description: This erratum is not relevant in a typical CAN network, with oscillator tolerances inside the specified limits, because an early start of frame condition (early-SOF) should not occur.

An early-SOF may only be a problem if the oscillators in the network operate at opposite ends of the tolerance range (maximum 1.58%), which could lead to a cumulated phase error after 10 bit-times larger than phase segment 2.

A corrupt ID will be sent out if a transmit message buffer is identified for transmission during INTERMISSION, and an early-SOF condition is entered due to a dominant bit being sampled during bit 3 of INTERMISSION.

The message sent will be taken from the newly set up transmit buffer (Tx MB), with the exception of the 1st 8 ID bits, which are taken from the previously selected Tx MB.

The CRC is correctly calculated on the resulting bit stream so that receiving nodes will validate the message.

The early-SOF condition is detailed in the Bosch CAN Specification Version 2.0 Part B, Section 3.2.5 INTERFRAME SPACING - INTERMISSION.

Workaround: 1) Configure Tx MBs during FREEZE mode, or

2) Out of FREEZE mode, configure Tx MBs during bus idle:

- For networks with low traffic, determine Bus Idle status by reading the Idle bit of the Error and Status register (CANx_ESR[IDLE]).

- For networks with high traffic, configure Tx MBs after the 3rd bit of intermission, and before the third bit of the CRC field from the next transmission.

e2685: FlexCAN: Module Disable Mode functionality not described correctly

Errata type: Errata

Description: Module Disable Mode functionality is described as the FlexCAN block is directly responsible for shutting down the clocks for both CAN Protocol Interface (CPI) and Message Buffer Management (MBM) sub-modules. In fact, FlexCAN requests this action to an external logic.

Workaround: In FlexCAN documentation chapter:

Section "Modes of Operation", bullet "Module Disable Mode":

Where is written:

"This low power mode is entered when the MDIS bit in the MCR Register is asserted. When disabled, the module shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules.."

The correct description is:

"This low power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU. When disabled, the module requests to disable the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules."

Section "Modes of Operation Details", Sub-section "Module Disable Mode":

Where is written:

"This low power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze Mode, it shuts down the clocks to the CPI and MBM sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit.."

The correct description is:

"This low power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze Mode, it requests to disable the clocks to the CAN Protocol Interface (CPI) and Message Buffer Management (MBM) sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit."

e551: FlexCAN: Transmit Buffers May Freeze / missing frame

Errata type: Errata

Description: If a received frame is serviced during reception of a second frame identified for that same MB (message buffer) and a new Tx frame is also initiated during this time, the Tx MB can become frozen and will not transmit while the bus is idle. The MB remains frozen until a new frame appears on the bus. If the new frame is a received frame, the frozen MB is released and will arbitrate for external transmission. If the new frame is a transmitted frame from another Tx MB, the frozen MB changes its C/S (control status word) and IFLAG to indicate that transmission has occurred although no frame was actually transmitted.

The frozen MB occurs if lock, unlock and initiate Tx events all occur at specific times during reception of two frames. The timing of the lock event affects the timing window of the unlock event as follows:

Situation A) Rx MB is locked during the 2nd frame. A frozen Tx MB occurs if:

1) Both of these events occur in either a-then-b or b-then-a order: a) A new transmission is initiated by writing its C/S between CRC3 (third bit of CRC field) and EOF7 (seventh bit of end of frame) of the 2nd frame.

b) The Rx MB is locked by reading its C/S after EOF6 of first frame and before EOF6 of second frame.

2) The Rx MB is unlocked between EOF7 and intermission at end of the second frame.

Notice in this situation that if the lock/unlock combination happens close together, the lock must have been just before EOF6 of the second frame, and therefore the system is very close to having an overrun condition due to the delayed handling of received frames.

Situation B) Rx MB was locked before EOF6 of the first frame; in other words, before its IFLAG is set. This is a less likely situation but provides a larger window for the unlock event. A frozen Tx MB occurs if:

1) The Rx MB is locked by reading its C/S before EOF6 of the first frame.

2) Both of these events occur in either a-then-b or b-then-a order: a) A new transmission is initiated by writing its C/S sometime between CRC3 and EOF7 of the second frame.

b) The Rx MB is unlocked between CRC3 and intermission at end of the second frame.

Notice in this situation that if the unlock occurs after EOF6, the first frame would be lost and the second frame would be moved to the Rx MB due to the delayed handling of received frames.

Situation C) Rx unlocked during bus idle. A frozen/missing Tx occurs if:

1) An Rx MB is locked before EOF6 of an incoming frame with matching ID and remains locked at least until intermission. This situation would usually occur only if the received frame was serviced after reception of a second frame.

2) An internal arbitration period is triggered by writing a C/S field of an MB.

3) The locked Rx MB is unlocked within two internal arbitration periods (defined below) before or after step 2).

4) 0xC is written to the C/S of a Tx MB within these same two arbitration periods. This step is optional if a 0xC was written in step 2).

Two internal arbitration periods are calculated as ((2 * number of MBs) + 16) bus clocks where the number of MBs can be reduced by writing to CAN_MCR[MAXMB]. Bus clocks are the high frequency bus clocks regardless of CAN_CR[CLK_SRC] setting.

Additional Notes:

1) The received frames can be transmitted from the same node, but they must be received into an Rx MB.

2) When the frozen Tx MB's IFLAG becomes set, an interrupt will occur if enabled.

3) The timestamp of the missing Tx will be set to the same timestamp value as the last reception before it was frozen.

4) If the user software locks the Rx MB before a frame is received, situation A can occur with a single received frame.

5) The issue does not occur if there were any additional pending Tx MBs before CRC3.

6) If multiple Tx MBs are initiated within the CRC3/EOF7 window (situation A and B) or two internal arbitration windows (situation C), they all become frozen.

Workaround: If received frames can be handled (lock/unlocked) before EOF6 of the next frame, situations A and C are avoided. If they are handled before CRC3, or lock times are below 23 CAN bit times, situation B is avoided.

If this cannot be guaranteed, situation A) and B) are avoided by inserting a delay of at least 28 CAN bit times between initiating a transmission and unlocking an Rx MB and vice-versa. Typically a system would use a mechanism to selectively add the necessarys delay. For example, software might use a global variable to record an external timer value (the FlexCAN timer can't be used as that would unlock) when initiating a new Tx or unlocking an Rx, and then add the required delay before performing the second action.

Situation C) can be avoided by inserting a delay of at least two internal arbitration periods between writing 0xC and unlocking the locked Rx MB.

e683: FlexCAN: receive time stamp may be incorrect

Errata type: Errata

Description: Unlocking a FlexCAN2 receive message buffer (MB) with pending frame sometime after a second frame is received or transmitted will cause capture of incorrect timestamp if the following conditions are satisfied:

1. A receive MB is locked via reading the Control/Status word, and has a pending frame in the temporary receive serial message buffer (SMB). The MB remains locked while receiving end of frame bit 6 of the pending frame.

2. The locked MB is unlocked anytime after the first bit of ID of another new frame is transmitted on the CAN bus. The new frame may be transmitted from the Flexcan2 or another CAN node.

In these conditions, the timestamp value for the unlocked receive MB will be the timestamp of the new frame and not the timestamp value from reception of the pending frame.

Workaround: Avoid locking MB during reception of EOF6 i.e. do not lock until corresponding IFLAG bit is set (prevents locking before present frame is received) and guarantee that received frames are handled (lock/unlocked) before EOF6 of the next frame.

e2424: FlexCAN: switching CAN protocol interface (CPI) to system clock has very small chance of causing the CPI to enter an indeterminate state

Errata type: Errata

Description: The reset value for the clock source of the CAN protocol interface (CPI) is the oscillator clock. If the CPI clock source is switched to the system clock while the FlexCAN is not in freeze mode, then the CPI has a very small chance of entering an indeterminate state.

Workaround: Switch the clock source while the FlexCAN is in a halted state by setting HALT bit in the FlexCAN Module Configuration Register (CANx_MCR[HALT]=1). If the write to the CAN Control Register to change the clock source (CANx_CR[CLK_SRC]=1) is done in the same oscillator clock period as the write to CANx_MCR[HALT], then chance of the CPI entering an indeterminate state is extremely small. If those writes are done on different oscillator clock periods, then the corruption is impossible. Even if the writes happen back-to-back, as long as the system clock to oscillator clock frequency ratio is less than three, then the writes will happen on different oscillator clock periods.

e1488: FlexCAN: writing to an active receive MB may corrupt MB contents

Errata type: Errata

Description: Deactivating a FlexCAN receive message buffer (MB) may cause corruption of another active receive MB, including the ID field, if the following sequence occurs.

1. A receive MB is locked via reading the Control/Status word, and has a pending frame in the temporary receive serial message buffer (SMB).

2. A second frame is received that matches a second receive MB, and is queued in the second SMB.

3. The first MB is unlocked during the time between receiving the CRC field and the 6th bit of end of frame (EOF) of the second frame.

4. The second MB is deactivated within 9 CPI clock cycles of the 6th bit of EOF, resulting in corruption of the first MB.

Workaround: Do not write to the Control/Status word after initializing a receive MB.

If a write (deactivation) is required to the Control/Status field of an active receive MB, either FREEZE the FlexCAN module or insert a delay of at least 27 CAN bit times plus 10 CPI clock cycles between unlocking one MB and deactivating another MB. This will avoid MB corruption, however frames may still be lost.

e817: JTAGC: EVTI and RDY require TCK to toggle

Errata type: Errata

Description: The Nexus/JTAG Read/Write Access Control/Status Register (RWCS) write (to begin a read access) or the write to the Read/Write Access Data Register (RWD)(to begin a write access) does not actually begin its action until 1 JTAG clock (TCK) after leaving the JTAG Update-DR state. This prevents the access from being performed and therefore will not signal its completion via the READY (RDY) output unless the JTAG controller receives an additional TCK. In addition, EVTI is not latched into the device unless there are clock transitions on TCK.

Workaround: The tool/debugger must provide at least one TCK clock for the EVTI signal to be recognized by the MCU. When using the RDY signal to indicate the end of a Nexus read/write access, ensure that TCK continues to run for at least 1 TCK after leaving the Update-DR state. This can be just a TCK with TMS low while in the Run-Test/Idle state or by continuing with the next Nexus/JTAG command.

Expect the affect of EVTI and RDY to be delayed by edges of TCK.

Note: RDY is not available in all packages of all devices.

e1666: JTAGC: SAMPLE instruction does not sample input data during board boundary scan testing.

Errata type: Errata

Description: Executing the SAMPLE instruction should take a snapshot of the input and output signals present at the pins, without interfering with the normal operation of the chip.

For pins configured as inputs, executing the SAMPLE instruction will result in the internally set, off value, of the pad being sampled and not the actual input value of the pin.

Workaround: Do not expect to sample input pin values when executing the SAMPLE or SAMPLE/PRELOAD instructions when using JTAG. Use the EXTEST and PRELOAD instructions to test connections between devices during boundary scan testing of boards.

e653: MPC5553: SIU_MIDR Revision field is 0x0010, NPC_DID[PIN]=0x53

Errata type: Information

Description: The part number field in the MCU Identification Register (SIU_MIDR[PARTNUM]) is 0x5553. The Mask revision number (SIU_MIDR[MASKNUM]) has been updated to 0x10. The Part Number Identification field in the Nexus Port Controller Device Identification Register/Message (NPC_DID[PIN]) is 0x53. The JTAGC and NPC DID revision was updated to 0x1 (NPC_DID[PN]=0x1).

Workaround: Software should be aware that the SIU_MIDR[MASKNUM] field can change in the future. Tools should be aware of the revision change in the JTAGC and the NPC Device Identification register/message.

e6726: NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled

Errata type: Errata

Description: The Nexus auxiliary message clock (MCKO) may be gated one clock period early when the MCKO frequency is programmed as SYS_CLK/8 in the Nexus Port Controller Port Configuration Register (NPC_PCR[MCKO_DIV]=111) and the MCKO gating function is enabled (NPC_PCR[MCKO_GT]=1). In this case, the last MCKO received by the tool prior to the gating will correspond to the END_MESSAGE state. The tool will not receive an MCKO to indicate the transition to the IDLE state, even though the NPC will transition to the IDLE state internally. Upon re-enabling of MCKO, the first MCKO edge will drive the Message Start/End Output (MSEO=11) and move the tool's state to IDLE.

Workaround: Expect to receive the MCKO edge corresponding to the IDLE state upon re-enabling of MCKO after MCKO has been gated.

e1957: NPC: MCKO_DIV can be set to 0x0 (1X MCKO)

Errata type: Information

Description: The Nexus Port Controller Port Configuration Register MCKO Divider bits (NPC_PCR[MCKO_DIV]) can be set to 0b000 to select a 1X clock rate as the Nexus Auxiliary output port frequency for the MCKO and MDO pins. Note: Depending on the system frequency, this may force the MCKO and MDO pins to switch at a frequency higher than can be supported by the pins. This maximum frequency is specified in the device electrical specification of the Nexus MCKO and MDO pins.

Workaround: Insure that the maximum operating frequency of the MDO and MCKO pins is not violated when setting the NPC_PCR[MCKO_DIV] values.

Note: tools may not support 1X mode. Check with your tool vendor.

e2096: NZ6C3: No sync message generated after 255 direct branch messages in history mode

Errata type: Information

Description: When using the branch history mode of direct branch program trace in the e200z6 core, a synchronization message is not transmitted after 255 program trace messages in a row as defined in the IEEE-ISTO 5001-2003 standard. This will occur if resource full messages are sent and not counted for triggering a sync message indicating that the branch history fields are full. The resource full message is generated when more than 31 direct branches occur without an indirect branch or exception.

Workaround: Debuggers should account for the possibility that more than 255 messages could be received without a program trace synchronization message by keeping track of the last known program trace address prior to branch history resource full messages.

e2058: NZ6C3: Branch Trace History field on PCM message is zero

Errata type: Errata

Description: The e200z6 nexus module (NZ6C3) branch history buffer contains the wrong reset value when the e200z6 core Nexus module gets reset (by going though the JTAG Test-Logic-Reset, assertion of the Test Reset Pin [if available], or re-enabling JCOMP (after de-assertion]). A reset value of 0x1 should be the register state after test reset, instead of 0x0, even if branch history mode is disabled.

Workaround: Ignore the Branch History field of all e200z6 core Program Correlation Message when branch tracing is NOT set to "history" mode (NZ6C3_DC1[PTM]=0b1).

e2103: NZ6C3: Data Trace of stmw instructions may cause overruns

Errata type: Information

Description: If Nexus data trace is enabled on a section of memory that is loaded or stored with a store multiple word (stmw), or load multiple word (lmw for data read traces), an overrun condition could occur, even if the stall on overrun feature is enabled (NZ6C3_DC1[OVC]=0b011). Stalls can only occur on instruction boundaries. The stmw/lmw instructions can generate up to 16 Nexus trace messages with a single instruction. If there are not 16 queue locations available, an overflow will occur. Stall mode does not stall the core until there are only four locations available in the e200 Nexus message queue. Therefore if a stmw/lmw generates more than four messages or if additional Nexus messages are generated, the queue will overflow. The stmw/lmw instructions load or store two 32-bit registers at a time (64-bit stores/loads) if an even number of registers are selected.

Workaround: If stall mode is enabled (NZ6C3_DC1[OVC]=0b011), limiting store multiple word instruction in a data trace region to store/load 8 registers or less, will improve the chances that an overrun will not occur, but this is dependent on other messages that could be generated simultaneously with the data trace messages. If stall mode is disabled, or stmw instructions with more than 8 registers are stored, accept overruns in the data and program trace flow.

e985: NZ6C3: Incorrect data traced on misaligned little endian store

Errata type: Errata

Description: The e200z6 core Nexus data trace of a misaligned (across 64-bit boundary) store to a little endian page will transmit the wrong data on the second half of the double word access.

The NSD submodule failed to account for the difference in byte lane selection required for misaligned little endian accesses. By selecting byte lanes without considering the possibility of the access being the second part of a misaligned transfer, incorrect data trace results occur on misaligned little endian stores.

Workaround: Either:

(1) Do not allow misaligned stores to little endian pages (use a compiler/linker switch if available) or

(2) Do not attempt to run data trace on little endian pages that are accessed with misaligned operations.

(3) Ignore the data on a missaligned 64-bit access that crosses a 64-bit word boundary on little endian memory pages.

e950: NZ6C3: Nexus stall mode may not prevent all Nexus overflow conditions

Errata type: Information

Description: Even if stall mode is enabled in the e200z6 Nexus Core Interface (NZ6C3 DC1[OVC]=0b011), the Nexus messages buffers could still overflow under certain instruction sequences due to the parallel processing capability of the e200z6 core. An overflow error message will be generated if this occurs.

Workaround: Reduce the amount of information being traced, or accept occasional gaps in trace information.

e1618: NZ6C3: No indication of an exception causing a Nexus Program Trace (PT) message as opposed to a retired branch instruction causing a PT message.

Errata type: Errata

Description: The e200z6 core Nexus (NZ6C3) transmits a Program Trace Indirect Branch message without indicating if the message was sent due to a taken branch or due to an exception. The instruction count for an exception is 1 less than a normal indirect branch. The result is that program trace reconstruction can be off by one instruction.

Workaround: Trace reconstruction tools should be aware that the I-CNT is different for Exceptions than for Indirect Branches. The tool may need to know (from the user or by parsing registers) the exception handler addresses from the Interrupt vector prefix register (IVPR) and the Interrupt vector offset registers (IVORxx). Users also should not jump directly to interrupt handler addresses. Tools can then differentiate between exceptions and indirect branches.

e2052: NZ6C3: RFM not sent for history buffer overflow caused by 'evsel'

Errata type: Errata

Description: The e200z6 Nexus should send a Resource Full Program Trace message (RFM) message when the predicate instruction history buffer overflows. The RFM message is implemented to avoid lost history information. A RFM is transmitted when the history buffer is full (contains 31 predicate instruction bits plus a stop bit) and either an isel instruction is executed or a conditional direct branch instruction is executed. However, if an evsel instruction is executed, which should also cause the RFM, it will not be transited and subsequent messages will contain corrupted history (HIST) information.

Workaround: Do not use the 'evsel' instruction, or don't use program trace "history mode" when tracing code segments containing an evsel instruction.

e1159: Pad Ring: ESD specifications are not met

Errata type: Errata

Description: Not all pins of the devices meet the ESD specifications of 2000 volts Human Body Model (HBM), specifically negative ESD from pins outside of the VRC domain referenced to either VRC33 or VRCCTL. 500V and 1KV HBM passed with a reduced number ESD events.

All pins meet the 500 volts (750 volts on corner balls) using the Field Induced Charged Device Model (CDM - per AEC Q100-002).

Workaround: Avoid exposure of VRC33 or VRCCTL pins to human body ESD events prior to mounting onto a printed circuit board (PCB). The PCB will provide protection after being mounted.

e1528: Pad Ring: Pin behavior during power sequencing

Errata type: Information

Description: The power sequence pin states table in the device data sheet (electrical specification) did not specify the influence of the weak pull devices on the output pins during power up. When VDD is sufficiently low to prevent correct logic propagation, the pins may be pulled high to VDDE/VDDEH by the weak pull devices.

At some point prior to exiting the internal power-on reset state, the pins will go high-impedance until POR is negated.

When the internal POR state is negated, the functional state during reset will apply and weak pull devices (up or down) will be enabled as defined in the device Reference Manual.

Workaround: The best solution is to minimize the ramp time of the VDD supply to a time period less than the time required to enable external circuitry connected to the device outputs.

e488: Pad Ring: Possible poor system clock just after POR negation.

Errata type: Information

Description: The pins RSTCFG_B and PLLCFG[0:1] select one of three PLL modes or allows a clock to be injected, bypassing the PLL. When Power On Reset (POR) negates, if the transitions on these pins selects the bypass mode, a poor clock on EXTAL can provide a poor clock to MCU logic no longer reset by POR. The state of that logic can be corrupted.

Workaround: If the default PLL and Boot configuration (external crystal reference and boot from internal flash) will be used, then negate the RSTCFG pin (=1). For any other configuration, depending on the final mode required, the pins must have the following values on the pins when the internal POR negates.

Final Mode RSTCFG_B PLLCFG[0] PLLCFG[1]

default 1 - -

external reference 0 1 1

external crystal - 1 -

dual controller - 1 -

After POR negates, the RSTCFG_B and PLLCFG[0:1] can be changed to their final value, but should avoid switching through the 0,0,0 state on these pins. See application note AN2613 "MPC5554 Minimum Board Configuration" for one example off the external configuration circuit.

e1969: Pad Ring: RSTOUT is 3-stated during the power-on sequence.

Errata type: Information
Description: RSTOUT_B is 3-stated during power on reset.

Workaround: Connect an external pull device to RSTOUT_B during power on reset. This should be pull-down unless an external reset configuration circuit is being used, in which case it should be pull-up. Refer to AN2613 'MPC5554 Minimum Board Configurations' for further information.

e3377: Pad Ring:Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge

Errata type: Errata
Description: The Nexus Output pins (Message Data outputs 0:15 [MDO] and Message Start/End outputs 0:1 [MSEO]) may drive an unknown value (high or low) immediately after power up but before the 1st clock edge propagates through the device (instead of being weakly pulled low). This may cause high currents if the pins are tied directly to a supply/ground or any low resistance driver (when used as a general purpose input [GPI] in the application).

Workaround: 1. Do not tie the Nexus output pins directly to ground or a power supply.

2. If these pins are used as GPI, limit the current to the ability of the regulator supply to guarantee correct start up of the power supply. Each pin may draw upwards of 150mA.

If not used, the pins may be left unconnected.

e1031: e200z6: Cache Invalidate/Clear Aborts due to masked core Interrupt

Errata type: Errata
Description: The e200z6 core will abort a "Cache Invalidate" (CINV) or "Cache Lock Bits Flash Clear" (CLFC) operation (in the Level 1 Cache Control and Status Register 0 (L1CSR0) if an external interrupt occurs during the operation, even if external interrupts to the core are disabled in the

Machine Status Register (MSR[EE]=0). The effect of this issue is that disabling external interrupts alone will not guarantee completion of a CINV or CLFC operation. Software must check for the cache abort condition.

Workaround: After clearing the EE bit in MSR, perform the following steps:

(1) Save the interrupt controller's current priority, INTC_CPR[PRI].

(2) Raise the interrupt controller's current priority to maximum (INTC_CPR[PRI]=0xF).

(3) Synchronize by executing msync and isync instructions.

(4) Ensure critical interrupts are also disabled (MSR[CE]=0).

(5) Execute CINV or CLFC cache operation.

After the cache operation completes, check to see if the abort bit has been set, indicating an external interrupt has caused the operation to terminate before completion. If set, clear the abort bit and retry the operation.

To restore normal operation after the cache invalidation or clear, perform the following steps:

(1) Restore the saved priority setting to INTC_CPR[PRI].

(2) Restore MSR[CE] if it was altered.

(3) Restore MSR[EE].

e2044: e200z6: Core renamed from e500z6

Errata type: Information
Description: The name of the main processing core has been changed from the e500z6 to the e200z6.

Workaround: Expect the new name for the e200z6 core in documentation.

e1143: e200z6: Data Storage Exception taken instead of Machine Check

Errata type: Errata
Description: A Data Storage exception (DSI) or Instruction Storage exception (ISI) will be taken if MSR[EE]=0. Instead the transfer error should have caused either a machine check or checkstop. The transfer error conditions that would cause either a DSI or ISI should only apply when MSR[EE]=1.

Workaround: Either: 1) Treat the DSI/ISI as an unrecoverable exception and do not return from exception handler, perform a reset instead. Or 2) If HID0[DCLREE]=1, check SRR1 in the DSI and ISI handlers; if SRR1[EE] bit is set then the exception will be recoverable and code can return from the interrupt without an issue. If the SRR1[EE] bit is clear then treat the exception as non-recoverable and code should not return from interrupt.

e846: e200z6: Debug interrupt (IVOR15) can be taken erroneously

Errata type: Errata
Description: If instruction complete debug events are enabled in the Debug Control Register 0 (DBCR0[ICMP] = 1) and an external interrupt causes a load multiple word, store multiple word, integer divide, or floating point divide instruction to be interrupted prior to completion, a debug

interrupt will be taken erroneously instead of the external interrupt. If external debug mode is enabled in the Debug Control Register 0 (DBCR0[EDM] = 1), debug mode will be entered after fetching the first instruction of the debug interrupt handler.

Workaround: Use the Go+NoExit method of single stepping in the OnCE Command Register (OCMD[GO] = 1, OCMD[EX] = 0) if external debug mode is enabled in the Debug Control Register 0 (DBCR0[EDM] = 1). If single stepping is implemented using instruction complete debug events, disable external interrupts (MSR[EE] = 0) prior to stepping over a load muliple word, store multiple word, integer divide, or floating point divide instruction.

e1536: e200z6: JTAG Part Identification is 0x2

Errata type: Information

Description: The Part Identification Number (PIN) in the e200z6 core JTAG and Nexus device identification messages and register (NZ6C3_DID) is 0x2. The complete e200z6 JTAG ID and DID is 0x07C0201D.

Workaround: Tools that use the e200z6 DID should expect updated values to identify the PPC core or use the complete device Nexus Port Controller DID message/register.

e914: e200z6: MFSPR may read prior value of SPEFSCR

Errata type: Errata

Description: If a move from the SPEFSCR special purpose register (mfspr spefscr) is performed IMMEDIATELY following an SPE instruction that modifies the integer overflow bits, the SPEFSCR will not have the latest values. Having ANY instruction between the SPE operation that modifies spefscr[SOVH,OVH,SOV,OV] bits and the mfspr spefscr will work correctly.

Workaround: At least one instruction must be placed between an instruction that updates the spefscr[SOVH,OVH,SOV,OV] bits and a mfspr spefscr. A NOP can be used if no other useful instruction is needed.

e1729: e200z6: MMU has 32 Table Entries

Errata type: Information

Description: Initial documentation for the MPC5554 stated that there would be only 24 table entries in the e200z6 core Memory Managment Unit (MMU). Actually, 32 entries were implemented and will remain in the future e200z6 devices.

Workaround: All 32 of the MMU table entries can be used.

e851: e200z6: wrong exception taken after FPU instruction preceding a lmw

Errata type: Errata

Description: When a floating point (fpu) instruction precedes a lmw instruction, or when there is one other instruction in between, and the fpu instruction has an fpu class exception, and an external or critical interrupt asserts while the lmw instruction is in the instruction pipeline, a wrong interrupt vector (IVOR) will be selected for exception processing.

Workaround: 1. Disable Floating Point Exceptions by clearing bits 25-29 of the Signal Processing Engine Status and Control Register (SPEFSCR - Default behavior).

2. Use compiler switches to avoid the use of the LMW/STMW instructions, these instructions are not typically used often.

3. Make sure two instructions are between an fpu instruction and a lmw class instruction.

e3421: e200z: Cache returns value, even when disabled

Errata type: Errata

Description: The cache line fill buffer of the e200z core will still provide a data cache hit, even if the Way Data Disable (WDD), Additional Ways Data Disable (AWDD), and the Way Access Mode (WAM, if available) bits are set.

Workaround: To avoid cache hit to the Line Fill Buffer after the WDD, AWDD, and WAM bits are disabled, a Cache Miss must be forced to clear the fill buffer. This can be done by branching to an instruction address that has not been fetched before.

e1136: eDMA: BWC setting may be ignored between 1st and 2nd transfers and after the last write of each minor loop.

Errata type: Errata

Description: The eDMA Transfer Control Descriptor Bandwidth Control field setting may be ignored between 1st and 2nd transfers and after the last write of each minor loop. This will occur if the source and destination sizes are equal. This behavior is a side effect of measures designed to reduce start-up latency. Reference Manuals may fail to mention this behavior.

Workaround: There are 2 possible workarounds:

1) Adjust the Transfer Control Descriptor (TCD) to make the source size not equal to the destination sizes (i.e. ssize = 16 bit, dsize = 32 bit). This delays the write which allows BWC[0:1] arriving from the TCD to be considered in the execution pipeline during start-up.

2) Adjust the TCD so the channel executes a single read/write sequence and then retires. In addition, the channel can be configured to execute a minor loop link to itself which will restart the channel after arbitration and channel start-up latency. The total number of bytes transferred can be controlled by the major loop count.

e2043: eDMA: Unexpected start of channel 15 during a spurious preemption

Errata type: Errata

Description: The standard product platform Direct Memory Access (DMA) module implements a preemption mechanism which allows a higher priority channel to temporarily suspend an executing DMA channel which is marked as preemptable.

The hardware is designed to detect and correct for a "spurious preemption" -- when the preempting channel's request is detected but goes inactive or is disabled before it can be started by the DMA engine. When a spurious preemption is detected, the DMA engine discards it and restores the original channel.

However, in some cases, a potential problem could occur when a higher priority hardware request in one channel group begins to preempt a lower priority channel in another group. There is a 1-cycle window where the problem can occur under very specific conditions:

1) If the preempting channel request is a hardware start request (not a software start bit request),

and

1) the preempting channel request is the only requesting channel within its group, and

2) if there is at least one other request in the same group as the active executing channel, and

3) if during the context switch, the higher priority hardware request is disabled via the DMA enable request register just before the DMA engine captures the new channel id, then instead of generating a spurious preempt request, the DMA may erroneously start channel 15 in the active channel's group.

Note: Since this problem involves an arbitration group change during the critical context switch, it only affects 32 and 64 channel configurations.

Workaround: To disable the hardware request of a high priority channel while a lower priority, preemptable channel is executing, perform one of the following options:

Option 1:

1. use the Disable Request (D_REQ) bit in the channel's Transfer Control Descriptor (TCD) to disable the channel after major loop completion.

Option 2:

1. disable all channels via the DMA Enable request high/low register (DMA_ERQRH/ DMA_ERQLH) or DMA Clear Enable Request Register (DMA_CERQR) bit 6 (clear all requests)

2. re-enable only the desired channels via the DMA_ERQ or the DMA Set Enable Register (DMA_SERQ).

Option 3:

1. ensure that one other request is active in the same group when disabling the channel; a request that is not the channel to be disabled

Option 4:

1. switch the group priority mode to round-robin in the DMA Control Register (DMA_CR)

2. disable the desired channel via DMA_ERQ or DMA_SERQ

3. switch the group priority mode back to fixed in the DMA_CR

e1168: eMIOS: Asynchronous reads of the A and B registers in IPM and IPWM modes may not be coherent

Errata type: Errata

Description: When using an eMIOS channel in Input period measurement (IPM) mode or input pulse width (IPWM) mode, asynchronous reads of the A and B registers may provide non-coherent values, resulting in either an incorrect period measurement (IPM mode) or an incorrect pulse width measurement (IPWM mode).

Workaround: The following pseudo code ensures coherent readings by accounting for CPU/eMIOS simultaneous access and checking the FLAG bit.

if (channel flag not set)

return

disable interrupts if enabled

read B(t0)

read A(t0)

read A(t1)

read B(t1)

clear channel flag

if [A(t0) - A(t1)] != 0

if [B(t0) - B(t1)] != 0

pulse width = A(t1) - B2(t1)

else

pulse width = A(t0) - B(t0)

else

pulse width = A(t1) - B(t1)

if (counter rollover)

adjust pulse width

enable interrupts if previously enabled.

e1747: eMIOS: Comparators A and B enabled by writing to A2 and B2 in DAOC mode

Errata type: Errata

Description: When the eMIOS is in Double Action Output compare (DAOC) mode, comparators A and B are enabled by the transfer of A2 to A1, and B2 to B1. However, writes to A2 and B2 before the transfers occur will also enable the comparators.

The main impact of this issue is that Output Update Disable (OUn) bit in the eMIOS Output Update Disable Register (eMIOS_OUDR) can not be used to cause both enables to occur at the same time. If the software sets OUn to disable the transfers and afterwards writes to A2/B2, the comparators will then be enabled. Then, if the timebase matches A1 or B1 (which have not yet been updated), the match will be recognized. The expected behavior is that the comparators should not be enabled until software clears OUn and the comparators are enabled simultaneously.

Workaround: When writing to the A2/B2 registers between the time that OUn is set and then cleared, the software should insure that the timebase does not match the old A1/B1 values.

e1834: eMIOS: OPMWC unable to produce close to 100% duty cycle signal

Errata type: Errata

Description: The Center Aligned Output Pulse Width Modulation with Dead-time Mode (OPWMC) of the eMIOS module does not function correctly if the trailing edge dead time is programmed to a value outside of the current cycle time. The OPWMC mode requires that matches occur in the specific order: A, A, and then B, where the first A must match on the up count of the modulus

counter, the second A match occurs on the down count of the modulus counter, and the B match occurs on the internal counter. If the programmed B match value is greater than the time required for the modulus counter to count down from the second A match and then up to the first A match of the next cycle, the first A match of the next cycle will be missed and the mode will not function correctly from that point on.

Workaround: Configure the selected modulus counter time base and the internal counter of the channel in OPWMC mode to count at the same rate. Program the value of the B match (dead time) to a value less than 2 times the programmed A match value.

e1201: eMIOS: Possible incoherent access in PEA mode

Errata type: Errata

Description: When reading the eMIOS channel A data register (EMIOS_CADRn) in Pulse/Edge Accumulation mode (PEA), it is possible that the data sampled will not be coherent with the value sampled at the next read of the channel B data register (EMIOS_CBDRn).

Workaround: If the eMIOS is used in PEA mode, after a read of the A and B data registers, if A is not greater than B, then discard this measurement and read both registers again after a new flag event.

e421: eMIOS: Possible incoherent accesses in IPWM/IPM modes

Errata type: Errata

Description: It is possible that reading the eMIOS Channel A Data Register (eMIOS_CADRx) and eMIOS Channel B Data Register (eMIOS_CBDRx) may result in incoherent data. This will occur when the CPU and eMIOS attempt to access the A register in the same clock.

In addition, if the application software does not check for the FLAG bit, there is a chance that the data from register B corresponds to an old measurement instead of the most recent. This is caused by the coherency mechanism, which locks B register updates once A register is read until B register is read.

Workaround: The following pseudo code ensures coherent readings by accounting for CPU/eMIOS simultaneous access and checking the FLAG bit.

```
if (channel flag not set)
return
disable interrupts if enabled
read B(t0)
read A(t0)
read A(t1)
read B(t1)
clear channel flag
if [A(t0) - A(t1)] != 0
if [B(t0) - B(t1) != 0
pulse width = A(t1) - B(t1)
else
```

pulse width = A(t0) - B(t0)

else

pulse width = A(t1) - B(t1)

if (counter rollover)

adjust pulse width

enable interrupts if previously enabled

e702: eMIOS: Writes to EMIOS_MCR register in IPM and IPWM modes may cause incoherent reads

Errata type: Errata

Description: When using the eMIOS in Input Period mode (IPM) or Input Pulse Width Mode (IPWM), writing the EMIOS_MCR register between reads of the A and B registers may result in incoherent values.

Workaround: When using IPM or IPWM modes, the software should not write to the EMIOS_MCR between reads of the A and B registers.

e1773: eQADC: conversions of muxed digital/analog channels close to the rail

Errata type: Information

Description: If the VDDEH9 and the VDDA power supplies are at different voltage levels, the input clamp diodes on the multiplexed digital and analog signals (AN12, AN13, AN14, and AN15) will clamp to the lower of the two supplies.

If VDDEH9 is lower than the VDDA, conversions on these channels will not obtain full scale readings if voltage is close the the VDDA voltage.

Workaround: When multiplexed digital/analog signals are used as analog inputs, connect VDDEH9 to VDDA and do not use any of the digital functions multiplexed on these pins.

e1013: eSCI : Automatic reset of the LIN state machine cause incorrect transmission

Errata type: Errata

Description: When the LIN FSM of the eSCI module performs an automatic reset due to a bus error condition, it is possible that the application is no longer synchronized to the LIN FSM. As a result, the eSCI may consider the payload data provided by the application via the LIN Transmit Register (eSCI_LTR) as a LIN frame header data and will generate an unexpected LIN frame.

Workaround: The application should disable the automatic LIN FSM reset functionality by setting LDBG bit in the LIN Control Register (eSCI_LCR) to 1.

e1381: eSCI: LIN Wakeup flag set after aborted LIN frame transmission

Errata type: Errata

Description: If the eSCI module is transmitting a LIN frame and the application sets and clears the LIN Finite State Machine Resync bit in the LIN Control Register 1 (eSCI_LCR1[LRES]) to abort the transmission, the LIN Wakeup Receive Flag in the LIN Status Register may be set (LWAKE=1).

Workaround: If the application has triggered LIN Protocol Engine Reset via the eSCI_LCR1[LRES], it should wait for the duration of a frame and clear the eSCI_IFSR2[LWAKE] flag before waiting for a wakeup.

e1221: eSCI: LIN bit error indicated at start of transmission after LIN reset

Errata type: Errata

Description: If the eSCI module is in LIN mode and is transmitting a LIN frame, and the application sets and subsequently clears the LIN reset bit (LRES) in the LIN Control register 1 (ESCI_LCR1), the next LIN frame transmission might incorrectly signal the occurrence of bit errors (ESCI_IFSR1[BERR]) and frame error (ESCI_IFSR1[FE]), and the transmitted frame might be incorrect.

Workaround: There is no generic work around. The implementation of a suitable workaround is highly dependent on the application and a workaround may not be possible for all applications.

e1017: eSCI: LIN fast bit error detection causes incorrect LIN reception

Errata type: Errata

Description: When using the eSCI module in LIN mode, if the fast bit error detection is enabled in the eSCI Control Register 2 (eSCIx_CR2[FBR]) and a bit distortion occurs on the RX line, the eSCI module may process the bit error signal incorrectly. This may cause unexpected transmission and reception behavior of the LIN state machine.

Workaround: The application should disable the fast bit error detection by setting the FBR bit to 0.

e2190: eSCI: LIN slave timeout flag STO not asserted if CRC is received too late

Errata type: Errata

Description: The eSCI module will not assert the Slave-Timeout Flag (STO) in the eSCI LIN Status Register 1 (LINSTAT1) if the checksum field of on RX frame is received later than the time given in the Timeout Value (TO) field of the LIN RX control header.

If the checksum field of on RX frame is not received at all, The STO flag will not be set and the LIN FSM will wait forever.

Workaround: The application should use a timer external to the eSCI module, that is started when a LIN RX frame transmission is started with an expiration time programmed to the expected duration of the reception. The timer is stopped, when the eSCI module signals the end of the LIN frame reception. If this timer expires, the eSCI has ran into the slave timeout condition. In this case, the application should clear the SCICR2[TE] and SCICR2[RE] bits to disable the transmitter and receiver, and should set the LINCTRL1[LRES] bit the reset the eSCI internal LIN slave and master tasks. To resume LIN frame data transmission the application should enable the transmitter and receiver and clear the LINCTRL1[LRES] bit to enable the LIN slave and master tasks.

e1614: eSCI: Low pulse on LIN RX line may prevent assertion of transmit data ready flag ESCI_SR[TXRDY]

Errata type: Errata

Description: If the eSCI module is in LIN mode and receives a low pulse on the RX line while transmitting a frame header or a stop bit, the eSCI internal LIN master and slave tasks may be stopped, and the transmit data ready flag ESCI_SR[TXRDY] will never be asserted again.

Each of the following scenarios of the RX low pulse may provoke this erroneous behavior.

a) The low pulse begins less than 12 bits before the start of the break character, and ends at least 21 bits and at most 30 bits after the start of the break character.

b) The low pulse begins at least 13 bits and at most 14 bits after the start of the break character, and ends at least 22 bits after the start of the break character.

c) The low pulse begins during the stop bit and has duration of at least 2 bits.

Workaround: The application should use a timer external to the eSCI module, that is started when a LIN frame transmission is started with an expiration time programmed to the expected duration of the transmission. The timer is stopped, when the eSCI module signals the end of the LIN frame transmission.

If this timer expires, the eSCI tasks have been stopped internally. In this case, the application should clear the Transmit Enable (SCICR2[TE]) and the Receive Enable (SCICR2[RE] bits to disable the transmitter and receiver, and should set the LIN FSM resync (LINCTRL1[LRES]) bit the reset the eSCI internal LIN slave and master tasks.

To resume LIN frame data transmission the application should enable the transmitter and receiver and clear the LINCTRL1[LRES] bit to enable the LIN slave and master tasks.

e1634: eTPU: LAST can fail on consecutive teeth

Errata type: Errata

Description: If the eTPU Angle Counter (EAC) enters High Rate mode with Tooth Program Register Last Tooth Indicator asserted (TPR[LAST]=1), and LAST is written to a one (1) again during High Rate mode, it negates when the EAC goes to Normal mode. This prevents using LAST on two consecutive teeth transition service requests to reset the eTPU Counter Register 2 (TCR2) one physical tooth after the other.

Workaround: If TCR2 must be reset on consecutive physical tooth detections, assert LAST for the second reset on a match service set for a low TCR2 count, preventing LAST to be asserted in High Rate mode.

e644: eTPU: MISSCNT can fail on sequential physical teeth

Errata type: Errata

Description: If the eTPU Angle Counter (EAC) detects a physical tooth with a non-zero value in the Missing Tooth Counter (MISSCNT) field of the Tooth Program Register (TPR), and during high-rate mode MISSCNT is written with a non-zero value, MISSCNT resets at the end of the high-rate mode.

One example of this use case is when the tooth wheel has more than 3 consecutive missing teeth, and MISSCNT is re-written with a non-zero value to support the extra missing teeth.

Another possible use case can occur when transitioning from normal tooth signal to a backup signal coming from a cam signal, for example.

These examples are illustrative, and do not exhaust the possibilities for the occurrence of the situation described.

Workaround: If a non-zero value is written to TPR[MISSCNT] and another non-zero value must be written after a single physical tooth is detected, the second non-zero value write should coincide with a match service on the TCR2 value estimated for the tooth. This will avoid MISSCNT being written in high-rate mode.

e112: eTPU: Prescaler phase shift between TCR1 and TCR2

Errata type: Information

Description: The Timer Counter Register 1 (TCR1) and Timer Counter Register 2 (TCR2) prescalers are initialized at slightly different times when the Global Timebase Enable (GTBE) is enabled. A phase shift of up to 2 clocks (one microcycle) can occur when the Timer Counter Registers (TCR1 and TCR2) increment in common multiples of the prescaler ratio.

Workaround: Expect a phase shift of up to 2 system clocks (one eTPU microcycle) between TCR1 and TCR2. TCR1 increments before TCR2.

e1728: eTPU: STAC bus export may skip 1 count

Errata type: Errata

Description: If the eTPU Angle Clock (EAC) is enabled and exported on the Shared Time and Counter bus (STAC) then one count may be skipped on random occasions. This only happens when the EAC transitions from Halt or High-rate mode to normal mode and the integer part of the Tick Rate Register (TRR) inside the eTPU is equal to 1. This skip does not occur on the TCR2 bus internal to the eTPU engine generating the angle clock.

Workaround: Either (1) use only greater-than-or-equal comparisons on angle counts imported from the STAC bus; or (2) limit the TRR integer part to 2 minimum. If TRR(integer) = 1 is a needed rate for maximum performance, the new TRR limitation can be compensated by either:

(a) doubling the TCR1 rate (for instance halving the TCR1 prescaler division), or

(b) halving the number of ticks per tooth (sacrificing angle accuracy).

e380: eTPU: TCR2, LAST can negate early in High Rate mode

Errata type: Errata

Description: If the eTPU Angle Counter (EAC) negates the Tooth Program Register (TPR) LAST bit in Normal mode and , without going through Halt, goes into High Rate mode later, the EAC can incorrectly reset the Timer Channel 2 (TCR2) and/or negate LAST at the end of High Rate mode. TCR2 is wrongly reset if LAST was negated when it entered High Rate mode. LAST is wrongly negated if it was asserted during High Rate mode. The TPR[LAST} bit indicates that the last tooth of the tooth wheel has occurred.

Workaround: Clear the Tooth Program Register LAST bit (TPR[LAST]=0) in the thread that services the transition of the first tooth of the wheel, in other words, the one that negates LAST. This operation is redundant with respect to TPR value, but fixes the EAC state to avoid an error.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
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

Document Number: MPC5553REVAE
Rev. 30 APR 2013