Freescale Semiconductor
Device Errata

MPC5200E
Rev. 5, 12/2011

MPC5200 (L25R) Errata

by: Microcontroller Division

This document identifies implementation differences between the MPC5200, L25R (Rev. 1.2), and the processor's description contained in the *MPC5200 Microcontroller User's Manual* and the *MPC5200 Microcontroller Data Sheet.* Refer to http://www.freescale.com for the latest updates.

Contents

freescale™ semiconductor

1 Overview

Table 1. MPC5200 (L25R) errata summary

Module	ID	Date Added	Title
ATA	158	01 July 2003	Section 2.1, ATA interrupt is not affected by FIFO errors
	292	01 July 2003	Section 2.2, ATA PIO write did not complete due to a LPC pipeline transaction
	325	01 July 2003	Section 2.3, ATA_ISOLATION glitch
BestComm	233	01 July 2003	Section 3.1, BestComm DMA handling of misaligned addresses
	329	10 May 2004	Section 3.2, BestComm Read Line Buffer does not invalidate
	351	01 July 2003	Section 3.3, BestComm DMA modules double initializing index registers
BDLC	369	05 Nov 2003	Section 4.1, BDLC BREAK Symbol timing differs from SAE J1850 specification
e300 Core	ERR 003383	01 Dec 2011	Section 5.1, If critical interrupt and normal interrupt are used in a system, the e300 core may hang
I^2C	416	19 Dec 2003	Section 6.1, I^2C Reaction to an arbitration loss is not I^2C compliant
JTAG/COP	330	01 July 2003	Section 7.1, LSRL corrupts XLB arbiter
LPC	343	01 July 2003	Section 8.1, ALE and TS changing on falling edge in 4:4:1 for LPC MUXED Mode
	347	01 July 2003	Section 8.2, 8-/16-bit accesses in MOST/GRAPHICS mode inoperable
	352	01 July 2003	Section 8.3, ALE to CS isolation clock is required for LPC MUXED mode
MSCAN	103	01 July 2003	Section 9.1, MSCAN filter bug for extended IDs
	415	05 Nov 2003	Section 9.2, MSCAN clock source option oscillator clock does not work
	499	01 Dec 2011	Section 9.3, Corrupt ID may be sent in Early-SOF condition
	MUCts 01373	01 Dec 2011	Section 9.4, Message erroneously accepted if bus error in bit 6 of EOF
PCI	224	19 Dec 2003	Section 10.1, PCI BestComm FIFO collision CPU vs.BestComm at low granularity
	310	01 July 2003	Section 10.2, Limited support of non-contiguous combinations of byte enables
	319	19 Dec 2003	Section 10.3, TEA not asserted when Maximum Retries limit reached
	322	19 Dec 2003	Section 10.4, Violation of PCI Tval min time
	348	01 July 2003	Section 10.5, No target aborts on target read errors
	429	19 Dec 2003	Section 10.6, PCI Bus must always be driven (bus parking)
	433	02 Mar 2004 (modified 10 May 2004)	Section 10.7, PCI arbiter can grant external and internal PCI master at the same time
	A11	19 Dec 2003	Section 10.8, External PCI bridge livelock

Table 1. MPC5200 (L25R) errata summary (continued)

Module	ID	Date Added	Title
PSC	363	01 July 2003	Section 11.1, SPI slave mode - first data must be ignored
	350	01 July 2003	Section 11.2, PSC3 pin muxing mode 0x111x does not work
	364	01 July 2003	Section 11.3, PSC UART — No return from break state
	370	05 Nov 2003	Section 11.4, PSC Limitation of Codec/I^2S slave mode
	447	10 May 2004	Section 11.5, PSC CODEC slave transmitter fail at DTS1=0 and ClkPol=1
SDRAM Controller	340	01 July 2003	Section 12.1, Self Refresh control in Deep Sleep Mode
	342	01 July 2003	Section 12.2, CS access (339)
SPI	349	01 July 2003	Section 13.1, SPIF bits set early for SPI clock < 200 kHz operation
USB	356	01 July 2003	Section 14.1, USB register addresses for HccaFrameNumber and HccaPad1 are swapped

2 ATA

2.1 ATA interrupt is not affected by FIFO errors

2.1.1 Description

FIFO error flags do not generate an ATA interrupt to the CPU.

2.1.2 Workaround

To identify a FIFO error, the FIFO status register must be polled.

2.2 ATA PIO write did not complete due to a LPC pipeline transaction

2.2.1 Description

There is an issue with the latency of the ATA to request access to the shared Address/Data bus. If there is an ATA PIO access (write or read) followed by any other XLB_PCI/LPC transaction that asserts a request and then receives a grant before the ATA request is asserted, the ATA does not the access to the shared Address/Data bus.

2.2.2 Workaround

Turn off XLB pipelining before performing ATA PIO transactions.

2.3 ATA_ISOLATION glitch

2.3.1 Description

The ATA_ISOLATION output is an active high signal to control external ATA transceiver devices and to isolate the ATA bus from the Local Plus (shared) bus. The ATA_ISOLATION is driven low after the positive edge of HRESET for 4 PCI_CLK cycles.

2.3.2 Workaround

This is a warning only.

3 BestComm

3.1 BestComm DMA handling of misaligned addresses

3.1.1 Description

BestComm DMA can produce misaligned word addresses on its Slave and Comm bus. These accesses are processed incorrectly by the Magenta write, Comm bus and SRAM interfaces which expect aligned address and appropriate byte selects asserted.

3.1.2 Workaround

Do not use misaligned accesses.

3.2 BestComm Read Line Buffer does not invalidate

3.2.1 Description

When the BestComm reads data from the XLB memory bus, it reads it in blocks of 32 bytes. If the BestComm is requesting sequential addresses with lengths of less than 32 bytes, an optimization is utilized which allows the BestComm to do a single access to the XLB memory bus and then read data from one of four Read Line Buffers (RLB) between the BestComm and the XLB.

For instance, if the BestComm reads 4 bytes from address 0x2340, then the BestComm will fetch data from 0x2340–0x235F from the XLB memory bus and leave it in the RLB. A subsequent read of four bytes at address 0x2344 will not result in another fetch from the SDRAM controller, but will instead be taken directly from the RLB.

The problem occurs in the previous example when a write to address 0x2344 occurs from a XLB master, say the CPU, in between the two reads. In this case, no check is made to see if the data is dirty and the data

in the RLB is still taken without a subsequent fetch from the SDRAM controller. Obviously, this is probably not the correct data.

3.2.2 Workaround

1. 1. The simplest workaround for this problem is to make another non-sequential read. For instance, read address 0x2340, then read address 0x2500, then read address 0x2344. This forces the BestComm to make three separate fetches. Note that this workaround is not as far fetched as it sounds when the usage of the BestComm is taken into account.
2. 2. A 2nd workaround is to force a non-sequential read and put the buffers in memory with at least 32 bytes between them.
3. 3. Another workaround is to use the buffers in reverse order. Transmit C then B then A. This will force the BestComm to do a non-sequential read at the beginning of each buffer.

3.3 BestComm DMA modules double initializing index registers

3.3.1 Description

A task code of the form :

```
LCDEXT:idx2 =idx3;idx2 >var13;idx2 +=inc0
LCD:idx3 =*(idx1 +var00000015);;idx3 +=inc1
```

creates problems with context switching. It is caused by two different modules within BestComm DMA initializing idx2 and idx3 twice and producing an incorrect result in idx2. This code is very rare and could be caused by an unsupported coding style in the tasks.

3.3.2 Workaround

Changing the order of idx definition in the previous LCD changes the problematic LCD to:

```
LCDEXT: idx2 = idx2; idx2 > var13; idx2 += inc0
LCD: idx3 = *(idx1 + var00000015); ; idx3 += inc1
```

which resolves the problem.

4 BDLC

4.1 BDLC BREAK Symbol timing differs from SAE J1850 specification

4.1.1 Description

The BREAK symbol is defined by the SAE J1850 spec Rev May2001 as follows:

- TX min ≥ 280 µs

- TX nom = 300 µs
- TX max µ≤ 5 ms
- RX min > 239 µs
- RX max ≤ 1 s

The MPC5200 BDLC provides a BREAK Tx nom time of 240 us, which is smaller than TX min ≥ 280 µs. This is a violation.

The SAE J1850 spec does not define the 4X mode.

4.1.2 Workaround

To transmit a BREAK symbol of the right length, re-adjust the BDLC prescaler by writing the corresponding value to the BDLC Rate Select Register.

5 e300 Core

5.1 If critical interrupt and normal interrupt are used in a system, the e300 core may hang

5.1.1 Description

If critical interrupt and normal interrupt are being used in a system, the e300 core may hang.

5.1.2 Workaround

Instead of using an RFI at the end of every exception handler, replace the RFI with the following:

Disable critical interrupts by setting MSR[CE] to 0 with **mtspr**.

Copy SRR0 and SRR1 to CSRR0 and CSRR1, respectively.

Execute an RFCI. This enables MSR[CE] and any other bits that original RFI would have set, including MSR[EE]

Sample code:

```
// Disable MSR[CE]
mfmsr   r2
lis     r3, 0xffff
ori     r3, r3, 0xff7f
  and     r2, r2, r3
  sync
  mtmsr   r2
  isync

// Copy SRR0, SRR1 to CSRR0 and CSRR1
  mfspr   r2, srr0
```

```
  mfspr   r3, srr1
  mtspr   csrr0, r2
  mtspr   csrr1, r3

… restore GPRs

rfci
```

6 I²C

6.1 I²C Reaction to an arbitration loss is not I²C compliant

6.1.1 Description

The I²C bus specification (Philips Semiconductors) defines: “A master which loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration.”

When the MPC5200 I²C controller as master loses the arbitration, it generates one extra clock pulse on the serial clock (SCL) line.

This becomes visible only if there is no other master driving the clock line. Potentially this becomes a problem for another node, which is may be forced by the clock pulse to an undefined state. This is a rare scenario, but a violation of the I²C specification.

6.1.2 Workaround

Make sure that the design of the external I²C bus system supports a recovery mechanism for the bus, like continuously toggling the SCLK line.

7 JTAG/COP

7.1 LSRL corrupts XLB arbiter

7.1.1 Description

When JTAG is scanning through the G2_LE core LSRL or boundary scan the XLB arbiter can be confused. Because of the XLB logic confusion, resume after scanning is not possible. This only affects JTAG debug operation.

7.1.2 Workaround

The sequence is:

1. Halt the core (SOFT STOP required).

2. Find the value of MBAR using the TLM_MBAR_SEL instruction.
3. Read and save off the XLB Arbiter Configuration register at MBAR + 0x1F40. This EXMEM sequence should NOT have bit 24 set when running the RunN counter.
4. Write the XLB Arbiter Configuration register at MBAR + 0x1F40 with 0x00000000. This EXMEM sequence should NOT have bit 24 set when running the RunN counter. This enables pipelining and turns off arbiter time out counters. Enabling pipelining allows the arbiter to reset itself when it gets a bad state. Disabling the time out counters avoids the problem of receiving TEAs when accessing unimplemented memory space on the MPC5200.
5. From here, all LSRL activity is permitted and External memory can be accessed. Any external memory accesses here should have bit 24 set. Setting this bit for these accesses gets around bad unimplemented memory accesses.
6. Restore the XLB Arbiter Configuration register at MBAR + 0x1F40 using EXMEM Write. Be sure bit 24 is set for this access.
7. Now you can resume.

NOTE

A general rule for setting bit 24 is if pipelining is disabled in the XLB arbiter configuration register, then do not set bit 24.

8 LPC

8.1 ALE and TS changing on falling edge in 4:4:1 for LPC MUXED Mode

8.1.1 Description

When running a multiplexed LPC transaction in 4:4:1 XLB:IP:PCI clock ratio, the ALE is asserted an extra half PCI clock before it deasserts and TSb asserts.

8.1.2 Workaround

Use the falling edge of ALE to latch addresses.

8.2 8-/16-bit accesses in MOST/GRAPHICS mode inoperable

8.2.1 Description

In the MOST/GRAPHIC mode, 8-bit or 16-bit XLB accesses cannot be handled.

NOTE

The issue is related to MOST/GRAPHICS mode only where the 2 bits TSIZ[1:0] are available at pins GPIO_WKUP_7 and TEST_SEL_1. The MUXed Mode is not impacted, as it provides 3 bits TSIZ[2:0] on EX_AD[30:28].

8.2.2 Workaround

Do not use 8-bit or 16-bit XLB access in MOST/GRAPHIC mode.

8.3 ALE to CS isolation clock is required for LPC MUXED mode

8.3.1 Description

For the Muxed mode, an isolation clock cycle is needed between the deassertion of ALE and the assertion of CS. ALE and CS are changing at the same clock edge. This can cause the “Data” and not the “Address” to be latched.

8.3.2 Workaround

Use the falling edge of ALE to latch addresses.

9 MSCAN

9.1 MSCAN filter bug for extended IDs

9.1.1 Description

MSCAN extended ID rejected if stuff bit between ID16 and ID15. For 32-bit and 16-bit identifier acceptance modes, an extended ID CAN frame with a stuff bit between ID16 and ID15 can be erroneously rejected, depending on IDAR0, IDAR1, and IDMR1.

Extended IDs (ID28-ID0) that generate a stuff bit between ID16 and ID15 are:

```
IDAR0 IDAR1 IDAR2 IDAR3
******** ***1111x xxxxxxxx xxxxxxxx
    where x = 0 or 1 (don’t care)
        * = pattern for ID28 to ID18 (see following)
Affected extended IDs (ID28 - ID18) patterns:
a) xxxxxxxxx01 exceptions:
        00000000001
        01111100001
        xxxx1000001 except 11111000001
b) xxxxx100000 exception: 01111100000
c) xxxx0111111 exception: 00000111111
d) x0111110000
```

```
e) 10000000000
f) 11111111111
g) 10000011111
```

When an affected ID is received,an incorrect value is compared to the 2nd byte of the filter (IDAR1 and IDAR5, plus IDAR3 and IDAR7 in 16-bit mode). This incorrect value is the shift register contents before ID15 is shifted in (i.e., right shifted by 1).

9.1.2 Workaround

If the problematic IDs cannot be avoided, the workaround is to mask certain bits with IDMR1 (and IDMR5, plus IDMR3 and IDMR7 in 16-bit mode).

Example 1: to receive the message IDs

```
xxxx xxxx x011 111x xxxx xxxx xxxx xxxx
```

IDMR1 etc. must be 111x xxx1,i.e., ID20, 19, 18, 15 must be masked.

Example 2: to receive the message IDs

```
xxxx 0111 1111 111x xxxx xxxx xxxx xxxx
```

IDMR1 etc. must be 1xxx xxx1, i.e., ID20 and ID15 must be masked.

In general, using IDMR1 etc.1111 xxx1, i.e., masking ID20, 19, 18, SRR, 15 hides the problem.

9.2 MSCAN clock source option oscillator clock does not work

9.2.1 Description

There is a CAN Clock Source Select feature in the MSCAN, controlled by the CLKSRC bit of the CANCTL1 register.

By default (reset), the IP bus clock is used as clock source for the MSCAN. Selecting the Oscillator clock (SYS_XTAL_IN) as clock source for the MSCAN does not work.

9.2.2 Workaround

Only the default IP bus clock can be used as clock source for the MSCAN.

9.3 Corrupt ID may be sent in Early-SOF condition

9.3.1 Description

This erratum is only relevant for applications using more than one transmit buffer and with oscillators operating at opposite ends of the tolerance range. A corrupt ID is sent out if a Tx message with highest priority is set up for transmission during bit 3 of INTERMISSION and a dominant bit is sampled leading to an early-SOF condition. The message sent is taken from the newly setup Tx buffer with the exception of the first eight ID bits taken from the originally selected Tx buffer. The CRC is correctly calculated on

the resulting bit stream so that receiving nodes validate the message. In a typical CAN network, with oscillator tolerances inside of the specified limits, this is not an issue because an early SOF condition should not occur. This may be a problem if the oscillators operate at opposite ends of the tolerance range (max. 1.58%), which could lead to a cumulated phase error after 10 bit times larger than phase segment 2.

NOTE

The CAN protocol condition referred to as early SOF in this erratum is detailed in a note in the “Bosch CAN Specification Version 2.0 Part B”, section 3.2.5 INTERFRAME SPACING – INTERMISSION.

9.3.2 Workaround

Use only one transmit buffer and do priority sorting in software. If more than one transmit buffer must be used, one of the following workarounds can be chosen:

- Do not release higher priority messages than those already scheduled (either determined by local priority or if equal, by hard-coded buffer priority) before all buffers are empty.
- Abort messages before scheduling higher priority transmissions.
- Prevent bad IDs passing acceptance filters by not assigning IDs (x) consisting of a combination of other assigned IDs (y,z), i.e. IDx[11:0] != {IDy[11:3], IDz[2:0]} for standard and IDx[28:21] != {IDy[28:21],IDz[20:0]} for extended identifiers.
- Allocate dedicated data length codes (DLC) to every ID and check for correspondence after reception.

9.4 Message erroneously accepted if bus error in bit 6 of EOF

9.4.1 Description

If a particular error condition occurs within the end of frame segment (EOF) of a CAN message, the MSCAN module recognizes and accepts a non-valid message as being valid, contrary to the CAN specification. The MSCAN module incorrectly validates messages after five recessive bits of the end of frame instead of after six bits. If a bus error occurs during the sixth bit of end of frame, the MSCAN module will already have accepted the message as valid, even although an error frame is transmitted and the receive error counter is incremented.

The CAN protocol states that message validation differs between bus transmitter and receiver devices (refer to part B, section 5 of CAN protocol for details). In the case where the seventh bit of the EOF segment is dominant, the message is valid for the receiver but not for the transmitter. This erratum extends this case to the sixth bit of the EOF segment.

9.4.2 Workaround

This erratum will not be an issue if the application software is protected against the known double receive problem of the CAN protocol. This problem occurs when a message is not recognized as valid by the transmitter, but is recognized as valid by a receiver, as described above. When this happens, the message is re-transmitted and hence the receiver will receive the same message twice.

10 PCI

10.1 PCI BestComm FIFO collision CPU vs.BestComm at low granularity

10.1.1 Description

When the FIFO granularity is 0-3 (not when it is set to 4 or higher) read data can be corrupted with no error indication because the PCI controller writes to the same location that the BestComm is reading from at the same time.

10.1.2 Workaround

Do not use a FIFO granularity below 4 (0-3).

10.2 Limited support of non-contiguous combinations of byte enables

10.2.1 Description

The Command/Byte Enable bus C/BE#[3:0]is driven by the initiator. In each data phase the byte enables indicate the bytes to be transferred within the currently-addressed double-word. When MPC5200 is the PCI target, some combinations of byte enables (the non-contiguous cases) are not correctly supported.

For external master writes with non-contiguous byte enables (active low) 0x2, 0x4, 0x5, 0x6, 0xa, if address bit 2 is low (corresponding to dh on XLB), the second beat of write data is put on the wrong data bus (dl). Non-contiguous PCI to XLB bus transfers require two XLB bus accesses. Refer to the *MPC5200 Microcontroller User's Manual*, PCI chapter. The incorrect supported combinations of byte enables are:

```
C/BE# PCI BUS AD
3:0 31:24 23:16 15:8 7:0
---------------------------
0010 OP3 OP2 --- OP1
0100 OP3 --- OP2 OP1
0101 OP3 --- OP2 ---
0110 OP3 --- --- OP2
1010 --- OP3 --- OP2
```

As a result, wrong data are written, no error wis indicated, and no target abort is issued.

10.2.2 Workaround

Do not use these combinations of non-contiguous byte transfers.

NOTE

Refer to the *MPC5200 Microcontroller User's Manual*, PCI chapter: The XLB bus supports misaligned operations; however, it is strongly recommended that software attempt to transfer contiguous code and data where possible. Non-contiguous transfers degrade performance.

10.3 TEA not asserted when Maximum Retries limit reached

10.3.1 Description

When the implemented retry counter reaches the retry count limit (Maximum Retries) no error will be sent back on XLB (Transfer error ack (tea_b) indicates that a bus error occurred).

Reaching the Maximum Retries limit generates a normal interrupt.

10.3.2 Workaround

Use the normal interrupt, which alerts the processor that a problem exists.

10.4 Violation of PCI Tval min time

10.4.1 Description

The PCI Timing Specification defines the timing parameter Tval (CLK to Signal Valid Delay):

Tval min 2 ns, max 11 ns (33 MHz)

Tval min 2 ns, max 6 ns (66 MHz)

When M66EN is asserted, the minimum specification for Tval(min) may be reduced to 1 ns if a mechanism is provided to guarantee a minimum value of 2 ns when M66EN is deasserted.

MPC5200 provides Tval min of 1 ns (instead of 2 ns), which is a violation of the Timing Specification for 33 MHz.

10.4.2 Workaround

Make sure that this reduced hold time budget is considered by the design of the external PCI bus system.

10.5 No target aborts on target read errors

10.5.1 Description

PCI target reads are supposed to respond with target aborts when a tea_b occurs on the internal xlb bus (e.g., a read from an unallocated DRAM space). Instead, when an transfer error occurs, the target path sends back fake data (Zeros) and completes the transaction (If the PCI 16/8 latency rule is not set).

Target-Abort is the termination mechanism that allows the target to terminate a transaction in which a fatal error has occurred, or to which the target will never be able to respond. Target-Abort will cause the initiator to end the transaction with no data transfer and no retry. Target-Abort is indicated to the initiator by simultaneously asserting STOP# and deasserting TRDY# and DEVSEL#.

10.5.2 Workaround

If the PCI16/8 latency rule is set, the target path issues at least a retry disconnect.

10.6 PCI Bus must always be driven (bus parking)

10.6.1 Description

The PCI Local Bus Specification (2.2) defines that the point-to-point and shared 32-bit PCI bus signals do not require pull-ups; bus parking ensures their stability.

When the bus is idle and no bus masters are requesting ownership, either the bus arbiter or a master that has the bus parked on it must enable its AD, C/BE#, and PAR output drivers.

Due to the nature of the MPC5200 PCI arbiter and the complexity of handling the shared PCI bus, parking on an external PCI master is not possible without incurring system instability. Therefore, the PCI arbiter will never leave the grant asserted to an external PCI master after a transaction has completed.

Parking on an internal non-PCI master (LPC, SCLPC, ATA) can violate the rule to drive the AD, C/BE#, and PAR pins.

10.6.2 Workaround

There is no real workaround. To avoid that the PCI bus is left floating for a long period dummy accesses to the LPC can be performed.

10.7 PCI arbiter can grant external and internal PCI master at the same time

10.7.1 Description

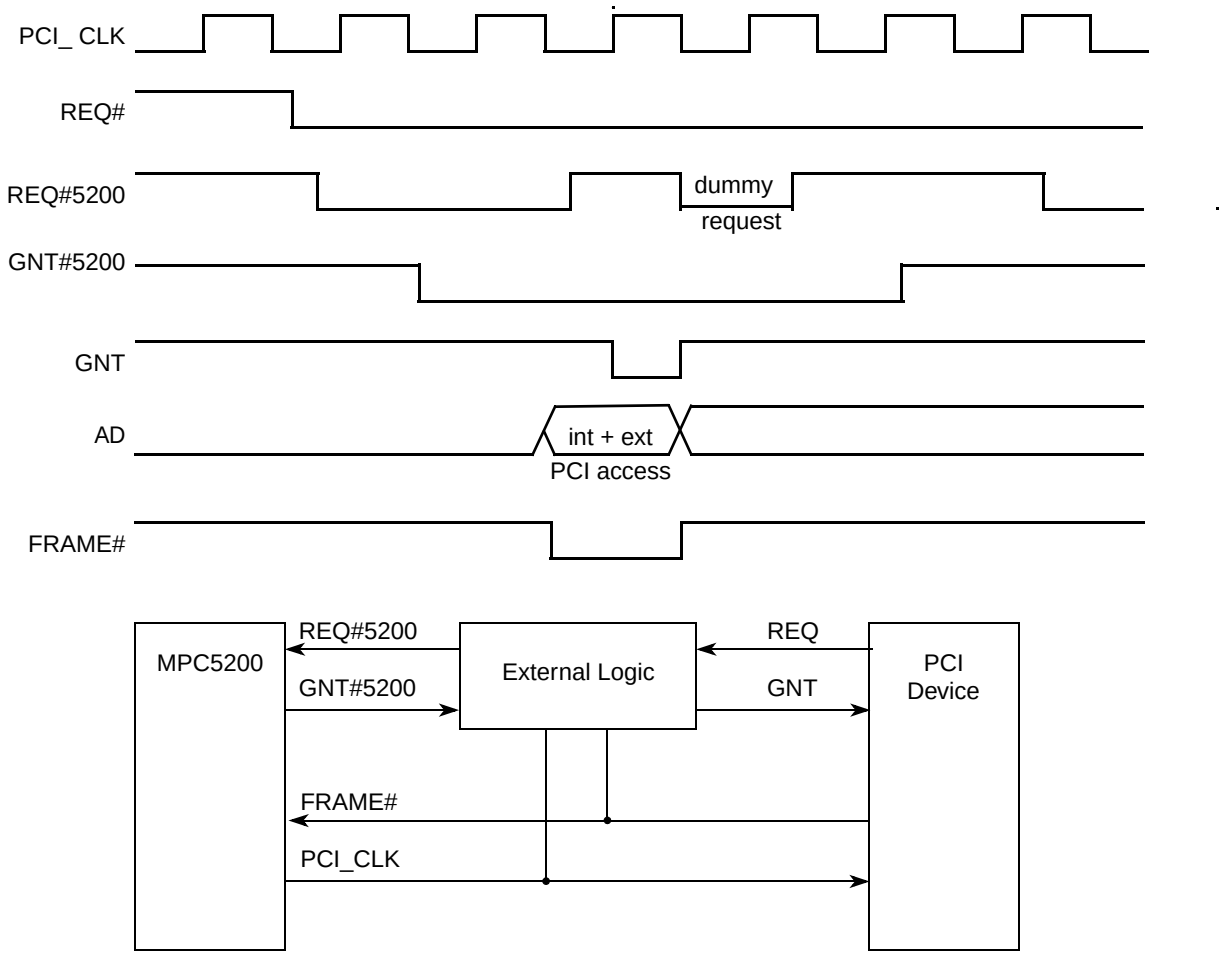
The internal PCI arbiter should grant bus ownership only to one master at a time.

This rule of operation is violated under certain conditions. An external PCI Initiator requests (REQ#) and gets the bus (GNT#). If exactly at this time, in a window of one IP bus clock cycle, an internal PCI Initiator asserts a request for the bus too, the internal PCI arbiter grants the internal requestor, in addition to the external one. As a result of two granted initiators there is contention on the PCI bus.

10.7.2 Workaround

External logic is required to delay the GNT# by at least two PCI clock cycles and to deassert the REQ# when FRAME# becomes active and to reassert REQ# again.

The following diagrams illustrate the workaround:



10.8 External PCI bridge livelock

10.8.1 Description

An external PCI device that is both a master and a slave can potentially cause system livelock.

Due to the nature of the MPC5200 PCI arbiter and the complexity of handling the shared PCI bus, an external PCI device that is both a master and a slave can cause system-wide livelock.

Specifically, this can only occur on such a device that has sideband communication between the master and slave partitions where the master can instruct the slave to retry inbound transactions until the master completes its transaction.

As an example, the MPC5200 may issue an I/O write to an external PCI device which is both a master and a slave. About the same time, this external master has a transaction that is pending, and is awaiting mastership of the PCI bus. Because of this, the external master instructs the slave to retry any inbound I/O writes until its outbound transaction is serviced. This is done because any inbound I/O write may change the state of the master which is trying to complete a transaction. In this case, the PCI arbiter first gives the grant to the core issuing the I/O write, but the transaction is retried by the external slave. Because of a limitation of the PCI arbiter, it can not rearbitrate incoming requests after the slaves first retry, and instead

continues to assert the grant to the core until its pending transaction is successfully completed. Thus the slave retries will occur forever, and no forward progress is made; the MPC5200 is in livelock.

Note that this situation will not always occur, as in the case when the outbound transaction on the XLB is placed in the PCI posted-write buffer. In this case, the XLB write transaction is completed, and the PCI arbiter will eventually arbitrate to the external device for bus mastership. However, if the XLB transaction is a read, or is not eligible or able to be placed in the posted-write buffer, then the system remains susceptible to livelock.

10.8.2 Workaround

There are two ways to avert this problem when using an external device that has a coupled master and slave component.

1. If possible, program the external slave to issue a Target Abort after excessive retries.
2. Program the MPC5200 PCI registers PCITTCR and PCIRTCR, Max_Retries field to catch slaves with excessive retries.

To alleviate the livelock, both options above require that an interrupt is generated to the CPU, and that the pending XLB transaction is aborted with a TEA (if it was not a posted write). The CPU interrupt is necessary to determine which XLB master was requesting the external slave so the transaction can eventually be reissued. The TEA is necessary to free up the XLB so the external device can finally obtain the grant from the PCI arbiter, thus alleviating the livelock.

The first option is the preferred work-around because it forces the external device to monitor its own retries for a specific transaction request. Once the retry counter expires, the slave issues a Target Abort instead of a retry, which causes a CPU interrupt and a TEA to be generated to the XLB.

The second option is similar to the first, except that the MPC5200 PCI controller monitors the retries for a given transaction to a particular slave. The Max_Retries counter is set in the PCI controller to a sufficiently large value (for example, 0xFF). When the Max_Retries counter expires because a transaction is constantly being retried, the CPU will receive an interrupt, and a TEA will be generated to the XLB.

11 PSC

11.1 SPI slave mode - first data must be ignored

11.1.1 Description

In SPI slave mode, the PSC module does not generate the bit clock. The bit clock will be generated by the master device and is an input to the PSC. The PSC requires some clocks for the configuration before any data can be transfered. The master provides the clock only together with the data.

Therefore, the first received and transmitted data are incorrect, the following transfers are correct.

11.1.2 Workaround

The first received and transmitted data must be ignored. The master must send a “dummy” data after each PSC-SPI configuration.

11.2 PSC3 pin muxing mode 0x111x does not work

11.2.1 Description

The pin muxing mode 0x111x for PSC3 does not work. In this mode the PSC3 group provides the signals for CODEC3 mode and SPI (from the stand-alone SPI module) at the same time. It is not possible to use this mode. The SPI part is working but the CODEC functionality is broken.

11.2.2 Workaround

When CODEC and SPI required, either use a different PSC for the CODEC, or set the SPI to use the Timer Group pins.

11.3 PSC UART — No return from break state

11.3.1 Description

The PSC UART cannot get back to the normal operation from the break state if the receiver was disabled.

11.3.2 Workaround

Do not reset RX during UART break state.

11.4 PSC Limitation of Codec/I^2S slave mode

11.4.1 Description

The PSC receiver in Codec or I^2S slave mode is not synchronized with the frame clock. Only the bit clock is used for the sampling of the serial data.

An inaccurate external (master) bit clock can force the receiver out of sync. There is no embedded Schmitt trigger for the external (master) bit clock input. In an noisy environment, a bit clock could be lost or a spike could be sampled.

In the case of an erroneous bit clock sequence, there is no indication of error and no workaround to bring the receiver back to sync. The receiver must be reset, the FiFo must be cleared, and the transmission must be started again.

11.4.2 Workaround

Make sure that a clean external (master) bit clock is available. An external Schmitt trigger can be added to improve the quality of the bit clock signal.

11.5 PSC CODEC slave transmitter fail at DTS1=0 and ClkPol=1

11.5.1 Description

The PSC CODEC slave transmitter mode does not work at the configuration DTS1 = 0 and ClkPol = 1. Incorrect data will be transferred. Affected are all CODEC (softmodem and I^2S) slave modes that use the DTS1 = 0 and ClkPol = 1 configuration.

11.5.2 Workaround

There is no workaround.

12 SDRAM Controller

12.1 Self Refresh control in Deep Sleep Mode

12.1.1 Description

The implemented S/W mechanism to control the SDRAM Self Refresh when entering and exiting Deep Sleep Mode does not work when the executed code is located in the external SDRAM.

12.1.2 Workaround

The portion of the code which controls the SDRAM Self Refresh before entering and after exiting

Deep Sleep Mode must be executed from Icache or external Flash memory.

12.2 CS access (339)

12.2.1 Description

There is problem when executing code from CS0 that writes data to CS1 of the SDRAM, and vice versa. As a result, exception 700 will be forced.

12.2.2 Workaround

Go through the standard method of calculating all command-to-command delays. When this is complete, determine the max of the parameters “srd2rwp,” “swt2rwp,” “brd2rp,” “brd2wt,” and “bwt2pre;” and set all of these parameters equal to that max value.

By setting all of these delay timer fields equal, all CSes are refreshed at the same time.

NOTE

All of the delay timer fields are not the same width. swt2rwp is only 3 bits, while all others are 4 bits. It may not be possible to set swt2rwp equal to the others. In this case there is still a vulnerability in the specific case of a sm1 single-write followed by an access to a different CS.

13 SPI

13.1 SPIF bits set early for SPI clock < 200 kHz operation

13.1.1 Description

At low clock frequency (SPI clock < 200 kHz), it looks like the SPIF bit is set in the SPI status register not after 8th SPI clock cycle, but a bit earlier (about 0.5 SPI_CLOCK). Writing new data into the SPI data register can cause a Write Collision (WCOL flag); i.e., the previous transfer is not yet completed.

13.1.2 Workaround

The problem can be solved by putting a delay before the next data writing.

14 USB

14.1 USB register addresses for HccaFrameNumber and HccaPad1 are swapped

14.1.1 Description

The Open Host Controller Interface (OHCI) specification describes the Host Controller Communication Area (HCCA), which is located inside the memory. The start address (base address) of that memory area is defined by the USB controller register HC Communication Register.

The HCCA includes the "virtual" registers HccaFrameNumber and HccaPad1. The offsets are 0x80 (for HccaFrameNumber) and 0x82 (for HccaPad1).

In the MPC5200 USB, these two virtual registers are swapped. The HccaFrameNumber is a copy of the Frame Number field at the USB HC Timing Reference Register.

14.1.2 Workaround

A the “C-level,” swap the two variables in the HCCA structure. The error occurs if an OHCI-specified compiled USB driver is used, because this HCCA does not have the right structure.

Table 2. Revision history

Rev. Number	Substantive Changes	Date of Release
0.0	Initial release.	07/2003
1.0	Added errata IDs 369 BDLC, 370 PSC, 415 MSCAN.	11/2003
1.1	Added errata IDs 319 PCI, 322 PCI, A11 PCI, 416 I^2C, 429 PCI.	12/2003
2.0	Added errata ID 433 PCI, removed 335.	03/2004
3.0	Added errata IDs 329 BestComm, 447, modified 433 PCI.	05/2004
4.0	Rebranded and reformatted.	08/2004
5.0	Added errata IDs ERR003383 e300 Core, 499 MSCAN, MUCts01373 MSCAN.	12/2011

How to Reach Us:

Home Page:
www.freescale.com

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

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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

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

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

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

Document Number: MPC5200E
Rev. 5
12/2011

freescale™ semiconductor