Advance Information

MPC107CE/D
Rev. 6, 5/2003

# MPC107 PCI Bridge/ Memory Controller Chip Errata



This document details all known silicon errata for the MPC107 and its derivatives. Table 1 provides a revision history for this chip errata document.

**Table 1. Document Revision History**

| Document Revision | Significant Changes |
|---|---|
| 0–2.7 | Earlier releases of document. |
| 3 | Added Errata 16. |
| 4 | Added Errata 17, 18, 19, and 20. |
| 5 | Updated document for erratum fixed by Rev. D (1.4) part. Fixed Errata 17, 18, 19, and 20). |
| 6 | Added Errata numbers 21 and 22. |

Table 2 provides a cross-reference to match the revision code in the processor version register to the revision level marked on the part.

**Table 2. Revision Level to Part Marking Cross-Reference**

| MPC107 Revision | Part Marking | Revision ID Register |
|---|---|---|
| 1.0 | Eng. | 0x10 |
| 1.1 | A | 0x11 |
| 1.2 | B | 0x12 |
| 1.3 | C | 0x13 |
| 1.4 | D | 0x14 |

Table 3 summarizes all known errata and lists the corresponding silicon revision level to which it applies. A ‘Y’ entry indicates the erratum applies to a particular revision level, while a ‘—’ entry means it does not apply.

**Table 3. Summary of Silicon Errata and Applicable Revision**

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 1 | ROM three-state wait functionality not working correctly for EDO/FPM systems | When the ROM three-state wait counter is programmed to a non-zero value for EDO/FPM systems, there exists a cycle window between the completion of a single-beat ROM read and the assertion of the three-state wait interval. During this window, the CCU can issue a DRAM transfer to the MCU in violation of the programmed three-state wait period. Potentially, this may result in a three-state collision on the memory data bus when the DRAM device subsequently tries to drive read data onto the bus before a slow ROM device relinquishes control of it. | Three-state collisions on the memory data bus for EDO/FPM memory systems with slow ROM, Flash, or PortX devices. | For EDO/FPM memory systems, program ROM three-state wait counter in memory configuration control registers to zero (that is; disable it). | Y | Y | — | — | — |
| 2 | TRIG_OUT pin not functioning correctly for JTAG operations | Some of the internal control signals on pin TRIG_OUT are connected in such a way which prevent results of certain JTAG sequences from being correct. | JTAG instructions related to the control of package pins will give incorrect results because the value on pin TRIG_OUT is not guaranteed to be correct. | Do not use JTAG instructions that control the value of external pins (for example; EXTEST). | Y | — | — | — | — |

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 3 | PCI latency timer expiration with target disconnect causes data corruption | The MPC107 as a master is doing a read or write on the PCI bus and its latency timer expires and the MPC107 grant is taken away, the MPC107 is forced off the PCI bus and will restart the next transaction the next time the MPC107 has access to the PCI bus. If, in the last beat of data transfer, a stop is asserted and the data is transferred, the MPC107 will put out the wrong address for the next transaction. The result is data corruption. This can happen for processor-to-PCI read/write or DMA read/write transactions on the PCI bus. | Data corruption can occur. | For processor-to-PCI transactions, increasing the latency value to a higher value (0x48 or more) will solve the problem.<br>For DMA-to-PCI operations, increasing the latency value to a higher value (0x48 or more) and setting the DMA mode register bit 23:22 to 0b01. Note that the DMA engine is operating with reduced PCI performance in this mode. | Y | Y | — | — | — |
| 4 | Unable to read register values for PCI_SERR and PCI_SERR enable | The read paths for PCI_SERR enable bit (bit 6 of error enabling register 2 <0xC4>) and PCI_SERR bit (bit 6 of error detection register 2 <0xC5>) are connected to logic 0. | Unable to read PCI_SERR bit (bit 6 of error detection register 2 <0xC5>) to determine if a PCI system error was detected. Also, unable to verify if the PCI_SERR enable bit (bit 6 of error enabling register 2 <0xC4>) is set. | The functionality of these register bits is still intact, so by enabling PCI_SERR enable, a PCI_SERR will still be detected. This PCI system error can still lead to the assertion of a machine check, but with a loss of visibility to the PCI_SERR detection bit. | Y | Y | — | — | — |

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 5 | Unable to perform direct configuration write accesses to PCI device in map A | A subset of the MPC107 configuration registers are accessible from the PCI bus through the use of PCI configuration cycles using either a direct or indirect access method. If map A is used and software attempts to perform a direct method configuration write to PCI devices, the transaction will come out to the PCI bus as an I/O write cycle. Note that direct method configuration reads from PCI devices functions properly. | Unable to perform direct configuration write to PCI devices in map A. | Software can use the indirect method configuration write access in map A to configure the PCI devices. | Y | Y | — | — | — |
| 6 | Read from 0xFFF00200–0xFFF00207 cannot negate an asserted $\overline{\text{MCP}}$ signal if PCI ROM is used | If the processor core detects an asserted $\overline{\text{MCP}}$ and this results in a machine check interrupt, the MPC107 will issue a code fetch from 0xFFF00200 if MSR[IP] = 1 or from 0x00000200 if MSR[IP] = 0. When seeing the fetch from the MCP exception handler, the MPC107 should then negate the $\overline{\text{MCP}}$ signal. However, if the address 0xFFF00200 is used while ROM is located in PCI memory space ($\overline{\text{RCS0}}$ = 0 during reset), this read operation will not negate the $\overline{\text{MCP}}$ signal. | Unable to negate $\overline{\text{MCP}}$ signal by opcode fetch from location 0xFFF00200 if PCI ROM is used. | The machine check exception handler can perform a dummy read from 0x00000200 in order to negate the $\overline{\text{MCP}}$ signal. This logic anomaly and work around results in the system software having control over when the $\overline{\text{MCP}}$ signal negates, where as it normally would not have control of this negation. | Y | Y | Y | Y | Y |

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 7 | SDMA0 held low at reset causes intermittent behavior | Pin SDMA0 should not be sampled at reset but is used by internal reset circuitry incorrectly. If SDMA0 is sampled as a logic 1 at reset, it performs two actions:<br>1. The DLL is reset internally via $\overline{\text{HRESET}}$.<br>2. The negation of $\overline{\text{HRESET\_CPU}}$ (output) to the microprocessor will occur $2^{17}$ clock cycles after $\overline{\text{HRESET}}$ (input) to the MPC107 is negated.<br>When SDMA0 is cleared to logic 0 during reset, this circuit behavior is intermittent and erratic. | Can result in intermittent behavior during hard reset. | Use external pull-up resistor on Rev. 1.1 = Rev. A devices. | Y | Y | — | — | — |
| 8 | $\overline{\text{GNT0}}$ has double output valid delay when the on-chip PCI arbiter is disabled | When the on-chip PCI arbiter is disabled, $\overline{\text{GNT0}}$ will function as the bus request from the MPC107 to the external PCI arbiter in order to get ownership of the PCI bus. However, due to the existence of a logic error, there is a double output valid delay for this output signal. This is caused by the logic having twice as many delay elements as required between the latch and the output pin. Note that these additional delay elements are controlled by PCI_HOLD_DELAY(0:2) bits (bits 6–4 of PMC register 2 <0x72>). | Systems using an external PCI arbiter may encounter a $\overline{\text{GNT0}}$ signal output valid timing problem (output valid time > 6.0 ns) when operating the PCI bus at 66 MHz, even if PCI_HOLD_DELAY(0:2) = 000 to, minimize the output valid time. | None | Y | Y | — | — | — |

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 9 | RFC refresh cycle for EDO/FPM DRAM is not working correctly in suspend mode | When the MPC107 is in the suspend mode with EDO/FPM DRAM, the MPC107 should be able to perform the memory refresh cycle (clocked by RFC clock input). However, due to the existence of a logic error, all the memory interface signals will be in a high-impedance state, and thus, prevent the MPC107 from being able to perform the refresh cycle to the EDO/FPM DRAM array. | Cannot support suspend mode with RFC refresh if EDO/FPM is used. | None | Y | Y | Y | Y | Y |
| 10 | PCI output valid time for MPC107 (Rev. 1.1 = Rev. A) not PCI 2.1 compliant for 66 MHz PCI operation | MPC107 (Rev. 1.1 = Rev. A) devices do not meet PCI 2.1 output valid specification for 66 MHz operation; the maximum output valid time specification is 6.0 ns for 66 MHz operation. Currently, MPC107 (Rev. 1.1 = Rev. A) devices have a 6.5 ns maximum output valid time. See MPC107 Bridge/Memory Controller Hardware Specifications, Table 10, Item 12a. | Systems with PCI devices requiring maximum PCI input setup times may not interface to the MPC107 (Rev. 1.1 = Rev. A) device operating at 66 MHz. | Be aware of the output valid timing constraints if using the MPC107 (Rev. 1.1 = Rev. A) devices in 66 MHz PCI applications. | Y | Y | — | — | — |
| 11 | DMA double-write on last PCI beat | For certain programmed values of the DMA DAR and BCR, the DMA controller will write the last beat of data out twice on the PCI bus. | The double-write may cause difficulty for FIFO-like structures on the PCI bus. No problem is expected for normal memory map devices on the PCI bus. | Software should try to avoid programming the DAR and BCR with the combination mentioned above if the PCI device has difficulty in receiving the double-write. | Y | Y | Y | — | — |

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 12 | MPC107 may misinterpret IDSEL during a PCI transaction that does not involve MPC107 | When a PCI master is performing a transaction to some other PCI device, the MPC107 may interpret the transaction as a configuration cycle to the MPC107. | This is a problem when the MPC107 is operating in peripheral (agent) mode and is a potential problem in host mode if the IDSEL pin of the MPC107 is not pulled down to a logic 0. | Motorola recommends, in host mode, to pull down the IDSEL input on the MPC107. In general, this is an acceptable solution for host mode operation as PCI configuration cycles are normally issued by the host controller.<br>In agent mode, external logic needs to be added to qualify the IDSEL input to the MPC107 in such a way where the IDSEL input will not be asserted after the address phase of any PCI transaction. This may introduce a timing problem as $\overline{\text{FRAME}}$ and IDSEL are involved in order to generate the qualified IDSEL signal to the MPC107 during the cycle when $\overline{\text{FRAME}}$ transitions from logic 1 to logic 0. | Y | Y | Y | — | — |

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 13 | DMA may execute incorrect PCI last beat because of PCI transactions not involving the MPC107 | For PCI transfers initiated by the MPC107 DMA, if the DMA transfer has one more beat to be completed (read or write) and is waiting for the PCI bus and another PCI master is currently transferring data to another PCI target, the DMA transfer for the last beat will not put out the correct cycle on the PCI bus the next time the MPC107 is granted the bus. | System may hang or data corruption can occur. | For a PCI memory target which can accept burst transfers without disconnecting at non-cacheline boundaries, the following programming steps should be taken for DMA transactions accessing PCI.<br>1. Do not program the DMA to perform only a single beat transfer.<br>2. Do not program the SAR/DAR registers to start at the last beat in a cache line and avoid programming the SAR/DAR and BCR with values such that (SAR/DAR + BCR) mod 0x20 = 1, 2, 3, or 4.<br>3.The MPC107 PCI latency timer should be programmed to 0x48 or higher and set DMA mode register reserved bits 23:22 to 0b01. Note that there will be a DMA performance reduction when operating in this mode, since the MPC107 will break up the DMA transaction by transferring a single cache line at a time on the PCI bus.<br>There is no work around for a PCI memory target which cannot handle burst transfers without disconnecting at non-cacheline boundaries (that is; single beat disconnects or random disconnects). | Y | Y | Y | — | — |

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 14 | PCI input high voltage for MPC107 not PCI 2.1 compliant | MPC107 devices do not meet the PCI 2.1 local bus specification minimum input high voltage ($V_{IH}$) DC electrical characteristic specification. The minimum PCI 2.1 input high voltage specification is 0.5 × $OV_{DD}$, where $OV_{DD}$ has a range of 3.0–3.6 V DC. Currently, MPC107 devices have a minimum input high voltage of 0.65 × $OV_{DD}$. | Systems with PCI devices capable of only driving the PCI specified minimum $V_{IH}$ (0.5 × $OV_{DD}$ ≤ min $V_{IH}$ < 0.65 × $OV_{DD}$) may not interface to the MPC107. | Ensure other PCI devices in the system with the MPC107 are capable of driving min $V_{IH}$ ≥ 0.65 × $OV_{DD}$. | Y | Y | Y | Y | Y |
| 15 | Data parity errors on 60x bus single beat writes are not detected | The MCP107 fails to detect the write parity errors and does not invoke a machine check error if all of the following conditions are met:<br>1. A single beat 60x write is being performed.<br>2. The CPU bus has corrupted data.<br>3. RMW parity mode is turned on (MCCR2[RMW_Par] is set).<br>4. Either ECC or parity modes are enabled. | This problem occurs on the MPC107 when working in ECC (where RMW has to be on) or normal parity modes, and CPU data gets corrupted in a single beat 60x write transaction. | Make all CPU-to-local memory writes which require error reporting burst writes (cacheable, write-back accesses). | Y | Y | Y | Y | Y |
| 16 | $\overline{MCP}$ signal of the MPC107 is driven with $OV_{DD}$ instead of $BV_{DD}$ | The $\overline{MCP}$ signal of the MPC107 had been documented in the MPC107 Bridge/Memory Controller Hardware Specifications, as being driven by $BV_{DD}$. This is not the case. The $\overline{MCP}$ signal is actually driven by $OV_{DD}$. | $\overline{MCP}$ is idle at 3.3 V. This must be noted especially for 2.5-V parts that connect with $\overline{MCP}$. | The MPC107 Bridge/Memory Controller Hardware Specifications, will be updated to reflect the correct supply voltage information for $\overline{MCP}$. | Y | Y | Y | Y | Y |

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 17 | Type 2 fast back-to-back transactions result in data corruption | If a PCI master issues a type 2 fast back-to-back transaction to the MPC107, the transaction results in data corruption. This is the case for both read and write transactions. | Type 2 fast back-to-back transactions are not supported on the MPC107. | Software should disable the ability to run fast back-to-back transactions on PCI master devices that can issue fast back-to-back transactions to the MPC107. | Y | Y | Y | Y | — |
| 18 | Chipset compatibility problems with MPC7450 | When the MPC107 asserts $\overline{TA}$ for a transaction to the MPC7450 before or at the time of the starting cycle of $\overline{AACK}$, an $\overline{ARTRY}$ issued by the next transaction is not recognized by the processor. | The $\overline{ARTRY}$ of the transaction that has the simultaneous $\overline{TA}$/$\overline{AACK}$ or an $\overline{ARTRY}$ of the next transaction will cause a problem. | None | Y | Y | Y | Y | — |
| 19 | PCI access latency to local memory | PCI accesses to local memory may stall if snooping is disabled PICR2[NOSNOOP_EN=1] because pipelined processor transactions have a priority 1.5 (Table 12-2 of the MPC107 PCI Bridge/Memory Controller User's Manual). | Increased access latency time for PCI and DMA transactions to or from local memory when snooping is disabled. | Enable snooping by clearing PICR2[NOSNOOP_EN].<br>If coherency is not otherwise required, the $\overline{GBL}$ signal can be pulled up at the processor. | Y | Y | Y | Y | — |
| 20 | Processor bus grant ($\overline{BG}$) lockout during PCI activity | If a **sync**/**eieio** instruction is issued to the MPC107 by the processor, the 60X bus arbiter will not grant the 60X bus to any processor until all the internal buffers are empty. | The MPC107 will not assert grants for transactions if the internal buffers are not available. | 1. On processors where address broadcast enable is provided, disable broadcasts of address-only transactions (HID0[ABE]=0).<br>2. Provide external logic which detects the assertion of an address-only broadcast and gates off the PCI grants to all PCI devices allowing a bubble to occur in the internal buffers. | Y | Y | Y | Y | — |





Freescale Semiconductor, Inc.

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

| No. | Problem | Description | Impact | Work Around | Silicon Revision | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 |
| 21 | MPC107 does not detect assertion of $\overline{\text{PERR}}$ signal for a certain case. | The MPC107 fails to detect the assertion of the PERR signal when the following conditions are met:<br>- The memory to PCI clock ratio is 2:1 or higher (for example, 2:1, 3:1 and 4:1)<br>- The MPC107 is the initiator of the PCI bus transaction<br>- A parity error occurs in the last data transfer of the transaction (for either single- or multiple-beat transactions) | The MPC107 fails to detect the assertion of $\overline{\text{PERR}}$ and does not report the parity error to the core via the internal $\overline{\text{mcp}}$ signal. Potentially corrupt data may be propagated. | None. | Y | Y | Y | Y | Y |
| 22 | MPC107 does not detect assertion of $\overline{\text{MCP}}$ signal for a certain doorbell register case in the messaging unit. | The MPC107 fails to detect the assertion of the $\overline{\text{MCP}}$ signal whether in host or agent mode when bit 31 of the Inbound Doorbell register (IDBR) is set even if the requirements for an MCP are met. | The MPC107 fails to detect an $\overline{\text{MCP}}$ in this case. | Use interrupts in the messaging unit to generate an $\overline{\text{MCP}}$ | Y | Y | Y | Y | Y |

## Error No. 1: ROM three-state wait functionality not working correctly for EDO/FPM systems

### Detailed Description:

When the ROM three-state wait counter is programmed to a non-zero value for EDO/FPM systems, there exists a cycle window between the completion of a single-beat ROM read and the assertion of the three-state wait interval. During this window, the central control unit (CCU) can issue a DRAM transfer to the memory control unit (MCU) in violation of the programmed three-state wait period. Potentially, this may result in a three-state collision on the memory data bus when the DRAM device subsequently tries to drive read data onto the bus before a slow ROM device relinquishes control of it.

### Projected Impact:

Three-state collisions on the memory data bus for EDO/FPM memory systems with slow ROM, Flash, or PortX devices.

### Work Arounds:

For EDO/FPM memory systems, program ROM three-state wait counter in memory configuration control registers to zero (that is; disable it).

### Projected Solution:

Book IV (Rev. 2.1), page 369 in the configuration register chapter, is revised to read that the three-state wait counter works only for SDRAM.

## Error No. 2: TRIG_OUT pin not functioning correctly for JTAG operations

### Detailed Description:

Some of the internal control signals on pin TRIG_OUT are connected in such a way which prevent results of certain JTAG sequences from being correct.

### Projected Impact:

JTAG instructions related to the control of package pins will give incorrect results because the value on pin TRIG_OUT is not guaranteed to be correct.

### Work Arounds:

Do not use JTAG instructions that control the value of external pins (for example; EXTEST).

### Projected Solution:

Fixed in Rev. 1.1 = Rev. A devices.

## Error No. 3: PCI latency timer expiration with target disconnect causes data corruption

### Overview:

The MPC107 as a master is doing a read or write on the PCI bus and its latency timer expires and the MPC107 grant is taken away, the MPC107 is forced off the PCI bus and will restart the next transaction the next time the MPC107 has access to the PCI bus. If, in the last beat of data transfer, a stop is asserted and the data is transferred, the MPC107 will put out the wrong address for the next transaction. The result is data corruption. This can happen for processor-to-PCI read/write or DMA read/write transactions on the PCI bus.

### Detailed Description:

The sequence of operation is as followed:

1. The MPC107 starts a transaction on the PCI bus and the MPC107 grant is taken away. The MPC107 latency timer expires and the MPC107 still has more data to transfer.
2. The MPC107 has to terminate the current transaction by deasserting $\overline{\text{FRAME}}$. $\overline{\text{IRDY}}$ is still asserted.
3. The target asserts $\overline{\text{STOP}}$ and $\overline{\text{TRDY}}$ for the last beat. Note that $\overline{\text{STOP}}$ is asserted only when $\overline{\text{FRAME}}$ is deasserted and continues for one PCI cycle. Data is transferred during this time.
4. The MPC107 finishes the current transaction and waits for the arbiter to grant the bus again.
5. Once grant is given to the MPC107, the MPC107 resumes the previously terminated transaction, but the address is incorrect.

### Projected Impact:

Data corruption can occur.

### Work Arounds:

For processor-to-PCI transactions, increasing the latency value to a higher value (0x48 or more) will solve the problem.

For DMA-to-PCI operations, increasing the latency value to a higher value (0x48 or more) and setting the DMA mode register bit 23:22 to 0b01. Note that the DMA engine is operating with reduced PCI performance in this mode.

### Projected Solution:

Fixed in Rev. 1.2 = Rev. B devices.

## Error No. 4: Unable to read register values for PCI_SERR and PCI_SERR enable

### Detailed Description:

The read paths for PCI_SERR enable bit (bit 6 of error enabling register 2 <0xC4>) and PCI_SERR bit (bit 6 of error detection register 2 <0xC5>) are connected to logic 0.

### Projected Impact:

Unable to read PCI_SERR bit (bit 6 of error detection register 2 <0xC5>) to determine if a PCI system error was detected. Also, unable to verify if the PCI_SERR enable bit (bit 6 of error enabling register 2 <0xC4>) is set.

### Work Arounds:

The functionality of these register bits is still intact, so by enabling PCI_SERR enable, a PCI_SERR will still be detected. This PCI system error can still lead to the assertion of a machine check, but with a loss of visibility to the PCI_SERR detection bit.

### Projected Solution:

Fixed in Rev. 1.2 = Rev. B devices.

## Error No. 5: Unable to perform direct configuration write accesses to PCI device in map A

### Detailed Description:

A subset of the MPC107 configuration registers are accessible from the PCI bus through the use of PCI configuration cycles using either a direct or indirect access method. If map A is used and software attempts to perform a direct method configuration write to PCI devices, the transaction will come out to the PCI bus as an I/O write cycle. Note that direct method configuration reads from PCI devices functions properly.

### Projected Impact:

Unable to perform direct method configuration write to PCI devices in map A.

### Work Arounds:

Software can use the indirect method configuration write access in map A to configure the PCI devices.

### Projected Solution:

Fixed in Rev. 1.2 = Rev. B devices.

## Error No. 6: Read from 0xFFF00200–0xFFF00207 cannot negate an asserted $\overline{\text{MCP}}$ signal if PCI ROM is used

### Detailed Description:

If the processor core detects an asserted $\overline{\text{MCP}}$ and this results in a machine check interrupt, the MPC107 will issue a code fetch from 0xFFF00200 if MSR[IP] = 1 or from 0x00000200 if MSR[IP] = 0. When seeing the fetch from the MCP exception handler, the MPC107 should then negate the $\overline{\text{MCP}}$ signal. However, if the address 0xFFF00200 is used while ROM is located in PCI memory space ($\overline{\text{RCS0}}$ = 0 during reset), this read operation will not negate the $\overline{\text{MCP}}$ signal.

### Projected Impact:

Unable to negate $\overline{\text{MCP}}$ signal by opcode fetch from location 0xFFF00200 if PCI ROM is used.

### Work Arounds:

The machine check exception handler can perform a dummy read from 0x00000200 in order to negate the $\overline{\text{MCP}}$ signal. This logic anomaly and work around results in the system software having control over when the $\overline{\text{MCP}}$ signal negates, where as it normally would not have control of this negation.

### Projected Solution:

The condition will remain unchanged because the work around seems to actually be an advantage in programming the chip.

## Error No. 7: SDMA0 held low at reset causes intermittent behavior

### Detailed Description:

Pin SDMA0 should not be sampled at reset but is used by internal reset circuitry incorrectly. If SDMA0 is sampled as a logic 1 at reset, it performs two actions:

1. The DLL is reset internally via $\overline{\text{HRESET}}$.
2. The negation of $\overline{\text{HRESET\_CPU}}$ (output) to the microprocessor will occur $2^{17}$ clock cycles after $\overline{\text{HRESET}}$ (input) to the MPC107 is negated.

When SDMA0 is cleared to logic 0 during reset, this circuit behavior is intermittent and erratic.

### Projected Impact:

Can result in intermittent behavior during hard reset.

### Work Arounds:

Use an external pull-up resistor on Rev. 1.1 = Rev. A devices.

### Projected Solution:

For Rev. 1.2 = Rev. B devices, SDMA0 is no longer sampled at reset. The DLL reset and negation delay of $\overline{\text{HRESET\_CPU}}$ will always occur.

## Error No. 8: $\overline{\text{GNT0}}$ has double output valid delay when the on-chip PCI arbiter is disabled

### Detailed Description:

When the on-chip PCI arbiter is disabled, $\overline{\text{GNT0}}$ will function as the bus request from the MPC107 to the external PCI arbiter in order to get ownership of the PCI bus. However, due to the existence of a logic error, there is a double output valid delay for this output signal.

This is caused by the logic having twice as many delay elements as required between the latch and the output pin. Note that these additional delay elements are controlled by PCI_HOLD_DELAY(0:2) bits (bits 6–4 of PMC register 2 <0x72>).

### Projected Impact:

Systems using an external PCI arbiter may encounter a $\overline{\text{GNT0}}$ signal output valid timing problem (output valid time > 6.0 ns) when operating the PCI bus at 66 MHz, even if PCI_HOLD_DELAY(0:2) = 000, to minimize the output valid time.

### Work Arounds:

None

### Projected Solution:

Fixed in Rev. 1.2 = Rev. B devices.

## Error No. 9: RFC refresh cycle for EDO/FPM DRAM is not working correctly in suspend mode

### Detailed Description:

When the MPC107 is in the suspend mode with EDO/FPM DRAM, the MPC107 should be able to perform the memory refresh cycle (clocked by RFC clock input). However, due to the existence of a logic error, all the memory interface signals will be in a high-impedance state, and thus, prevent the MPC107 from being able to perform the refresh cycle to the EDO/FPM DRAM array.

### Projected Impact:

Cannot support suspend mode with RFC refresh if EDO/FPM is used.

### Work Arounds:

None

### Projected Solution:

TBD

## Error No. 10: PCI output valid time for MPC107 (Rev. 1.1 = Rev. A) not PCI 2.1 compliant for 66 MHz PCI operation

### Detailed Description:

MPC107 (Rev. 1.1 = Rev. A) devices do not meet PCI 2.1 output valid specification for 66 MHz operation; the maximum output valid time specification is 6.0 ns for 66 MHz operation. Currently, MPC107 (Rev. 1.1 = Rev. A) devices have a 6.5 ns maximum output valid time. See the *MPC107 Bridge/Memory Controller Hardware Specifications,* Table 10, Item 12a.

### Projected Impact:

Systems with PCI devices requiring maximum PCI input setup times may not interface to the MPC107 (Rev. 1.1 = Rev. A) device operating at 66 MHz.

### Work Arounds:

Be aware of the output valid timing constraints if using the MPC107 (Rev. 1.1 = Rev. A) devices in 66 MHz PCI applications.

### Projected Solution:

Fixed in Rev. 1.2 = Rev. B devices; this errata only applies to Rev. 1.1 = Rev. A devices.

## Error No. 11: DMA double-write on last PCI beat

### Detailed Description:

For certain programmed values of the DMA DAR (destination address register) and BCR (byte count register), the DMA controller will write the last beat of data out twice on the PCI bus when the DMA engine is programmed either in local-to-PCI-memory or PCI-to-PCI-memory transfer mode. No data corruption occurs when this double-write happens, and the status register updates normally after the second beat write has completed. The combination of DAR and BCR that results in the double-write can be determined as follows:

1. (DAR + BCR) mod 0x20 = R, where R is a number between 0x00 and 0x1F.

   If R = 0x09–0x0C, 0x19–0x1C, or 0x11–0x14, then the double-write occurs.

   Example 1: DMA 42 (decimal) bytes from 0x0000_0000 to 0x8000_0000.

   R = (DAR + BCR) mod 0x20

   R = (0x8000_0000 + 0x2A) mod 0x20

   R = (2,147,483,648d + 42d) mod 32d

   R = 10d = 0x0A

   R = 0x0A, which is in the range of 0x09–0x0C; therefore, double-write of last beat to PCI will occur.

   Example 2: DMA 50 (decimal) bytes from 0x0009_0000 to 0x0009_4FE0.

   R = (DAR + BCR) mod 0x20

   R = (0x0009_4FE0 + 0x32) mod 0x20

   R = (610,272d + 50d) mod 32d

   R = 18d = 0x12

   R = 0x12, which is in the range of 0x11–0x14; therefore, double-write of last beat to PCI will occur.

### Projected Impact:

The double-write may cause difficulty for FIFO-like structures on the PCI bus. No problem is expected for normal memory map devices on the PCI bus.

### Work Arounds:

Software should try to avoid programming the DAR and BCR with the combination mentioned above if the PCI device has difficulty in receiving the double-write.

### Projected Solution:

Fixed in Rev. 1.3 = Rev. C devices.

## Error No. 12: MPC107 may misinterpret IDSEL during a PCI transaction that does not involve MPC107

### Detailed Description:

When a PCI master is performing a transaction to some other PCI device, the MPC107 may interpret the transaction as a configuration cycle to the MPC107.

This will happen during any cycle when:

1. $\overline{\text{FRAME}}$ is asserted, and
2. $\overline{\text{C/BE}}$[3:0] is either 0b1010 or 0b1011, and
3. IDSEL input to MPC107 is asserted, and
4. AD[1:0] = 0b00.

If above conditions occur while the MPC107 is the target of the transaction, the MPC107 will function as expected.

If the above conditions occur when the MPC107 is not the target of the transaction, the MPC107 will assert $\overline{\text{SERR}}$ if bit 8 of the PCI command register <0x04> is set. $\overline{\text{SERR}}$ will oscillate in this manner: $\overline{\text{SERR}}$ will be driven low for two clocks and then three-state for two clocks. The first occurrence of $\overline{\text{SERR}}$ asserting will be three clocks after the detected event.

Regardless of whether bit 8 of the PCI command register <0x04> is set, two cycles after the detected event, the MPC107 will always assert $\overline{\text{DEVSEL}}$ for one cycle. After the DEVSEL assertion, the internal state machine may lose state and may not respond to any accesses from an external PCI master.

### Projected Impact:

This is a problem when the MPC107 is operating in peripheral (agent) mode and is a potential problem in host mode if the IDSEL pin of the MPC107 is not pulled down to a logic 0.

### Work Arounds:

Motorola recommends, in host mode, to pull down the IDSEL input on the MPC107. In general, this is an acceptable solution for host mode operation as PCI configuration cycles are normally issued by the host controller.

In agent mode, external logic needs to be added to qualify the IDSEL input to the MPC107 in such a way where the IDSEL input will not be asserted after the address phase of any PCI transaction. This may introduce a timing problem as $\overline{\text{FRAME}}$ and IDSEL are involved in order to generate the qualified IDSEL signal to the MPC107 during the cycle when $\overline{\text{FRAME}}$ transitions from logic 1 to logic 0. (Refer to Figure 1 and Figure 2.)

### Projected Solution:

Fixed in Rev. 1.3 = Rev. C devices.

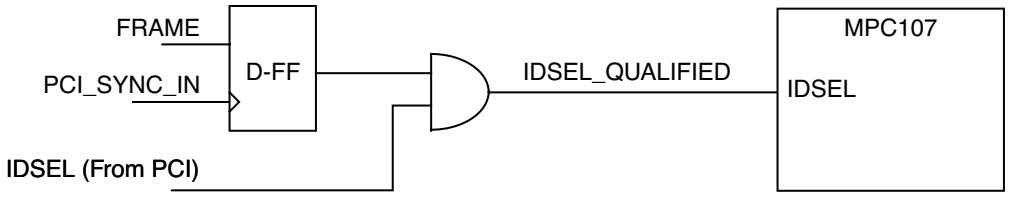
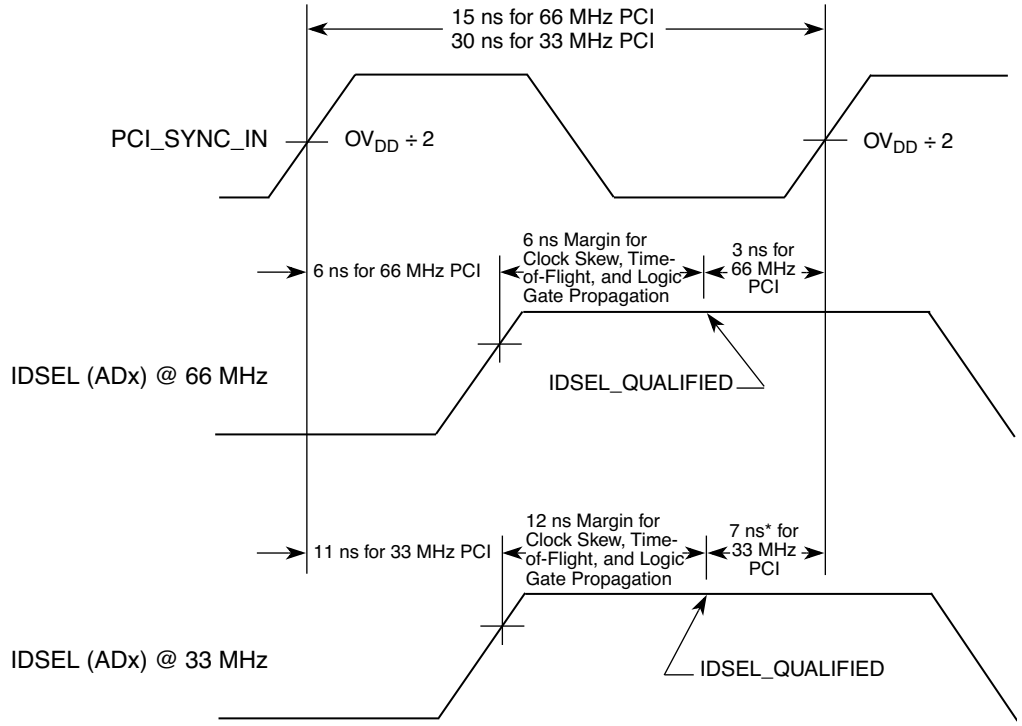



**Figure 1. External Logic Work Around for Agent Mode**




*MPC107 specifies 3 ns for PCI input setup; therefore, there are 4 ns of additional margin for logic gate propagation time.

**Figure 2. Timing Diagram for External Logic**

## Error No. 13: DMA may execute incorrect PCI last beat because of PCI transactions not involving the MPC107

### Detailed Description:

For PCI transfers initiated by the MPC107 DMA, if the DMA transfer has one more beat to be completed (read or write) and is waiting for the PCI bus and another PCI master is currently transferring data to another PCI target, the DMA transfer for the last beat will not put out the correct cycle on the PCI bus the next time the MPC107 is granted the bus.

The error condition is as follows:

1. The MPC107 DMA has one more beat to transfer on the PCI bus. The one more beat condition can be a result of either of the following:
   - The DMA is only performing a single beat transfer.
   - The DMA is currently requesting transfer of more than one beat (up to a cache line), and the PCI target issues a disconnect to the MPC107 such that the next time the MPC107 is granted the PCI bus, there is only one remaining beat to transfer.

   Note that the MPC107 may have more data to transfer afterward, since the size of each transfer (up to a cache line) depends on the DMA queue and number of bytes left to read or write. Also, disconnecting at a cache line boundary is acceptable.
2. Another PCI master currently has mastering access on the PCI bus and transfers data to a PCI target other than the MPC107 that is currently waiting to transfer one more beat on the PCI bus.

   Note that this errata does not occur if DMA has more than one beat waiting to be transferred when the interfering cycle occurs, or if the interfering cycle is targeted toward the MPC107 which has the pending DMA transfer.
3. Then, if the MPC107 is granted the bus it will try to transfer the last data beat.

   Because of a logic error triggered by step 2, the MPC107 will put out the wrong cycle on the PCI bus. During this incorrect transaction any/all of the following may occur:
   - $\overline{\text{C/BE}}$ signals can be driven to an incorrect value (0x0 or 0xF) during the address and data phases.
   - It is possible for the logic to reread the next to last beat even though it has already read it once.
   - The DMA status register may be incorrect (that is; the CB bit of the DSR is prematurely terminated and the interrupt prematurely activated) if the condition occurs on the very last data beat of a DMA PCI write transfer (that is; the DMA will finish transferring the last beat after the CB bit is prematurely cleared and the interrupt has been activated).

### Projected Impact:

System may hang or data corruption can occur.

### Work Arounds:

For a PCI memory target which can accept burst transfers without disconnecting at non-cacheline boundaries, the following programming steps should be taken for DMA transactions accessing PCI.

1. Do not program the DMA to perform only a single beat transfer.

2. Do not program the SAR/DAR registers to start at the last beat in a cache line and avoid programming the SAR/DAR and BCR with values such that (SAR/DAR + BCR) mod 0x20 = 1, 2, 3, or 4.
3. The MPC107 PCI latency timer should be programmed to 0x48 or higher and set DMA mode register reserved bits 23:22 to 0b01. Note that there will be a DMA performance reduction when operating in this mode, since the MPC107 will break up the DMA transaction by transferring a single cache line at a time on the PCI bus.

There is no work around for a PCI memory target which cannot handle burst transfers without disconnecting at non-cacheline boundaries (that is; single beat disconnects or random disconnects).

## Projected Solution:

Fixed in Rev. 1.3 = Rev. C devices.

## Error No. 14: PCI input high voltage for MPC107 not PCI 2.1 compliant

### Detailed Description:

MPC107 devices do not meet the PCI 2.1 local bus specification minimum input high voltage ($V_{IH}$) DC electrical characteristic specification. The minimum PCI 2.1 input high voltage specification is 0.5 ∞ $OV_{DD}$ where $OV_{DD}$ has a range of 3.0–3.6 V DC. See the *MPC107 Bridge/Memory Controller Hardware Specifications,* Table 3. Currently, MPC107 devices have a minimum input high voltage of 0.65 ∞ $OV_{DD}$.

### Projected Impact:

Systems with PCI devices capable of only driving the PCI specified minimum $V_{IH}$ (0.5 ∞ $OV_{DD}$ ≤ min $V_{IH}$ < 0.65 ∞ $OV_{DD}$) may not interface to the MPC107.

### Work Arounds:

Ensure other PCI devices in the system with the MPC107 are capable of driving min $V_{IH}$ ≥ 0.65 ∞ $OV_{DD}$.

### Projected Solution:

None

## Error No. 15: Data parity errors on 60x bus single beat writes are not detected

### Detailed Description:

The MCP107 fails to detect the write parity errors and does not invoke a machine check error if all of the following conditions are met:

1. A single beat 60x write is being performed.
2. The CPU bus has corrupted data.
3. RMW parity mode is turned on (MCCR2[RMW_Par] is set).
4. Either ECC or parity modes are enabled.

### Projected Impact:

This problem occurs on the MPC107 when working in ECC (where RMW has to be on) or normal parity modes, and CPU data gets corrupted in a single beat 60x write transaction. As a result, for ECC transactions, the corrupted data will be used to generate an ECC syndrome and both will be written to SDRAM without detection. For normal parity mode, the corrupted data will be written into the SDRAM and the parity byte will be recalculated for the entire line (with the corrupted data) and also written into SDRAM.

For CPU burst write transactions, the problem does not occur for either ECC or parity modes.

### Work Arounds:

Make all CPU-to-local memory writes which require error reporting burst writes (cacheable, write-back accesses). This work around may not be implemented for transactions that involve I/O devices which may not have caching capability.

### Projected Solution:

No plans to fix.

## Error No. 16: $\overline{\text{MCP}}$ signal of the MPC107 is driven with $OV_{DD}$ instead of $BV_{DD}$

### Detailed Description:

The $\overline{\text{MCP}}$ signal of the MPC107 had been documented in the *MPC107 Bridge/Memory Controller Hardware Specifications*, as being driven by $BV_{DD}$. This is not the case. The $\overline{\text{MCP}}$ signal is actually driven by $OV_{DD}$.

### Projected Impact:

$\overline{\text{MCP}}$ is idle at 3.3 V. This must be noted especially for 2.5-V parts that connect with $\overline{\text{MCP}}$.

### Work Arounds:

The *MPC107 Bridge/Memory Controller Hardware Specifications*, will be updated to reflect the correct supply voltage information for $\overline{\text{MCP}}$.

### Projected Solution:

No plans to fix.

## Error No. 17: Type 2 fast back-to-back transactions result in data corruption

### Detailed Description:

Type 2 fast back-to-back transactions are those that access multiple targets sequentially.

If a PCI master issues a type 2 fast back-to-back transaction to the MPC107, the transaction results in data corruption. This is the case for both read and write transactions.

Data that is read will be corrupted. Write transactions will write to incorrect locations or write bad data to a specified location.

### Projected Impact:

Type 2 fast back-to-back transactions are not supported on the MPC107.

### Work Arounds:

Software should disable the ability to run fast back-to-back transactions on PCI master devices that can issue fast back-to-back transactions to the MPC107. This can be done by clearing bit 9 of the PCI command register in the master device.

### Projected Solution:

Fixed in Rev. 1.4 = Rev. D. In this revision, bit 7 of the PCI status register (0x06) ia disabled. The bit is hardwired to 0, indicating that the MPC107 (as a target) is not capable of accepting fast back-to-back transactions.

## Error No. 18: Chipset compatibility problems with the MPC7450

### Overview:

The MPC107 has compatibility issues with the MPC7450.

### Detailed Description:

The MPC7450 implementation of the earliest transfer of data during an MPX/60x bus transaction is more restrictive than previous PowerPC™ processors including the MPC7400/MPC7410. This implementation creates a backwards compatibility issue with older system chipsets, including the MPC107.

In an MPC7450 system, the system chipset logic must ensure that the first (or only) assertion of $\overline{TA}$ for a data transfer does not occur sooner than the last cycle of the snoop response window (cycle after $\overline{AACK}$). Note that $\overline{DBG}$ can still be driven as early as $\overline{TS}$. Likewise, the system chipset logic must ensure that $\overline{TEA}$ is not asserted before the last cycle of the snoop response window.

When the MPC107 asserts $\overline{TA}$ for a transaction to the MPC7450 before or at the time of the starting cycle of $\overline{AACK}$, an $\overline{ARTRY}$ issued by the next transaction is not recognized by the processor.

### Projected Impact:

The $\overline{ARTRY}$ of the transaction that has the simultaneous $\overline{TA}$/$\overline{AACK}$ or an $\overline{ARTRY}$ of the next transaction will cause a problem. Any system that includes an MPC107 and MPC7450, where the $\overline{ARTRY}$ may be asserted, when the MPC107 asserts $\overline{TA}$ or $\overline{TEA}$ for a data tenure before or during the $\overline{AACK}$ for that transaction, may hang.

### Work Arounds:

None

### Projected Solution:

Fixed in Rev. 1.4 = Rev. D.

## Error No. 19: PCI access latency to local memory

### Detailed Description:

It is possible for the processor to prevent a PCI agent from accessing the SDRAM and thereby affect the minimum PCI and/or DMA access latency if the processor is performing a series of pipelined reads or writes. PCI accesses to local memory may stall if snooping is disabled PICR2[NOSNOOP_EN=1] because pipelined processor transactions have a priority 1.5 (Table 12-2 of the *MPC107 Bridge/Memory Controller User's Manual)*.

### Projected Impact:

Increased access latency time for PCI and DMA transactions to or from local memory when snooping is disabled.

### Work Arounds:

Enable snooping by clearing PICR2[NOSNOOP_EN]. This improves the PCI access latency to local memory by eliminating pipelined processor transactions that have a priority 1.5 in Table 12-2 of the *MPC107 Bridge/Memory Controller User's Manual*. Also, if coherency is not otherwise required, the $\overline{\text{GBL}}$ signal can be pulled up at the processor guaranteeing that the processor will never $\overline{\text{ARTRY}}$ the snooped transaction.

### Projected Solution:

Fixed in Rev. 1.4 = Rev. D.

## Error No. 20: Processor bus grant ($\overline{BG}$) lockout during PCI activity

### Detailed Description:

If a **sync**/**eieio** instruction is issued to the MPC107 by the processor, the 60X bus arbiter will not grant the 60X bus to any processor until all the internal write buffers are empty. These buffers are processor-to-PCI-write data buffers (PRPWB0 and PRPWB1), PCI-to-local-memory-write buffers (PCMWB0, and PCMWB1) and the copyback buffer.

It is possible for a PCI device to perform continuous accesses into SDRAM such that the internal buffers (PCMWB0, PCMWB1, and copyback) will not be empty. In such a case, the processor will be prevented from progressing beyond the **sync**/**eieio** address broadcast.

### Projected Impact:

System which cannot disable broadcasting of address-only **sync**/**eieio** instructions (for example; MPC7400/7410) or systems that require the broadcasting of these instructions (for example; MPX systems).

### Work Arounds:

1. On processors where address broadcast enable is provided, disable broadcasts of address-only transactions (HID0[ABE]=0).
2. Provide external logic which detects the assertion of an address-only broadcast and gates off the PCI grants to all PCI devices allowing a bubble to occur in the internal buffers.

### Projected Solution:

Fixed in Rev. 1.4 = Rev. D. In this revision, only the processor-initiated write buffers will need to be emptied out before MPC107 issues the bus grant to processor again.

## Error No. 21: MPC107 does not detect assertion of $\overline{\text{PERR}}$ signal for a certain case.

### Description

The MPC107 fails to detect the assertion of the $\overline{\text{PERR}}$ signal when all of the following conditions are met:

— The memory to PCI clock ratio is 2:1 or higher (for example, 2:1, 3:1 and 4:1)

— The MPC107 is the initiator of the PCI bus transaction

— A parity error occurs in the last data transfer of the transaction (for either single- or multiple-beat transactions)

### Projected Impact

The MPC107 fails to detect the assertion of $\overline{\text{PERR}}$ and does not report the parity error to the core via the internal $\overline{mcp}$ signal. Potentially corrupt data may be propagated.

### Solution

No plans to fix.

### Work-around

Use external logic to monitor the $\overline{\text{PERR}}$ signal and assert the NMI signal when a data parity error is detected on the last data transfer.

## Error No. 22: MPC107 does not detect assertion of $\overline{\text{MCP}}$ signal for a certain doorbell register case in the messaging unit.

### Description

The MPC107 fails to detect the assertion of the $\overline{\text{MCP}}$ signal whether in host or agent mode when bit 31 of the Inbound Doorbell register (IDBR) is set even if the requirements for an $\overline{\text{MCP}}$ are met.

The requirements for an $\overline{\text{MCP}}$ in this special case include:

HID0[EMCP]=1, PICR1[MCP_EN]=1, MSR[ME]=1

Set bit 31 of the IDBR, setting this bit while the IMIMR[DMCM]=0.

Even if the above conditions are met, an MCP does not occur assert long enough to be recognised by the processor.

### Projected Impact

The internal $\overline{\text{MCP}}$ signal is asserted for only one clock. Since the processor requires the internal $\overline{\text{MCP}}$ signal to be held asserted for at least two clocks, the MPC107 fails to detect the assertion of $\overline{\text{MCP}}$ in this case.

Apart from not getting an $\overline{\text{MCP}}$ for the above description, the MPC107 does not take a machine check exception even if enabled and bits 8,7 or 4 of the Inbound Message Interrupt Status Register are set (IMISR[8-7,4]=1).

### Solution

No plans to fix.

### Work-around

Use interrupts in the messaging unit to generate an $\overline{\text{MCP}}$.

**HOW TO REACH US:**

**USA/EUROPE/LOCATIONS NOT LISTED:**

Motorola Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-480-768-2130
(800) 521-6274

**JAPAN:**

Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu Minato-ku
Tokyo 106-8573 Japan
81-3-3440-3569

**ASIA/PACIFIC:**

Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre, 2 Dai King Street
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

**TECHNICAL INFORMATION CENTER:**

(800) 521-6274

**HOME PAGE:**

www.motorola.com/semiconductors






MPC107CE/D