PowerPC™

IBM27-82660 Revision 1.0
Errata Summary

MPR660ESU–00

This document details differences between the operation of the 660 Bridge chipset as described in the IBM27–82660 PowerPC to PCI Bridge User's Manual*, IBM document number MPR660UMU–01, and the actual operation of the Revision 1.0 chipset (IBM27–82663 Rev. 1.0 and IBM27–82664 Rev. 1.0). Where applicable, workarounds are described.*

April 19, 1995

IBM®

Document Revision History

Version 1.0 (April 19, 1995) Errata for the 1.0 revision of the chipset.

Part Identification, IBM27-82660 Chipset

IBM27-82663 (663 Buffer)
Chipset Rev. 1.0 contains IBM27-82663 Rev. 1.0. The package is marked 94G0235.

IBM27-82664 (664 Controller)
Chipset Rev. 1.0 contains IBM27-82664 Rev. 1.0. The package is marked 94G0232.

IBM27-82660 Errata Summary 1.0

For IBM27-82663 Rev. 1.0 and IBM27-82664 Rev. 1.0

April 19, 1995

All references to the 660 Bridge chipset, the 663 Buffer, and the 664 Controller refer to the revision 1.0 levels of these devices. All references to the *660 Bridge User's Manual* refer to the *IBM27–82660 PowerPC to PCI Bridge User's Manual*, IBM document number MPR660UMU–01, which is a beta level document.

Unless otherwise defined, the errata listed in this document detail differences between the operation of the 660 Bridge chipset as described in the *660 Bridge User's Manual* and the actual operation of the revision 1.0 chipset (IBM27–82663 revision 1.0 and IBM27–82664 revision 1.0).

Section 1 contains tips and workarounds for bringing up a system using the 1.0 revision of the 660 Bridge chipset. Errata are arranged by topic.

Section 2 lists the individual errata.

Section 3 describes the function and installation of two PLDs. These devices are used to implement the external logic that is required to correct various errata.

1 Tips and WorkArounds

This section contains tips and workarounds for bringing up a system using revision 1.0 of the 660 Bridge chipset. The following items are discussed:

- Software issues
- Recommended options and modes
- Hardware work arounds for revision 1.0 bugs
- Corrections to the example Implementation for the 660 Bridge
- Corrections to the beta version (MPR660UMU–01) of the 660 Bridge User's Manual

1.1 Software Issues

This section describes items which are important to the microcode, firmware, boot code, and potentially the operating system code.

1.1.1 660 Bridge Register Set Initialization

All of the registers in the 660 Bridge default to a usable value except for those which control the operation of the memory controller. Note that all error checking defaults to being masked (disabled). The memory controller registers must be initialized to correct values based on the type and quantity of memory installed (see the 660 Bridge User's Manual for details). Table 1 lists the registers that must be initialized to set up the memory controller:

Table 1. Memory Controller BCRs

Indexed BCR Name	Index
Memory Bank Starting Address	80h–87h
Memory Bank Extended Starting Address	88h–8Fh
Memory Bank Ending Address	90h–97h
Memory Bank Extended Ending Address	98h–9Fh
Memory Bank Enable	A0h
Memory Timing Register 1	A1h

Table 1. Memory Controller BCRs

Indexed BCR Name	Index
Memory Timing Register 2	A2h
Memory Bank Addressing Mode	A4h–A7h
Refresh Timer	D0h–D1h
Bridge Chip Set Options 3	D4h

Section 5.3.12 of the 660 Bridge User's Manual contains an example of how to program the memory controller.

- The Memory Bank Starting Address and Memory Bank Extended Starting Address BCRs must be set according to the size and number of banks of memory installed.
- The Memory Bank Ending Address and Memory Bank Extended Ending Address BCRs must be set according to the size and number of banks of memory installed. Note that the ending address for a given bank is the starting address for the next bank minus one (with the 20 least significant bits truncated). Also note that due to errata 25, the ending address of all unused banks must be set to zero.
- The Memory Bank Enable BCR must be set according to the number of banks of memory installed. Note that due to errata 18, the correlation between the bits of this register and the bank that they enable is reversed from what is stated in the 660 Bridge User's Manual.
- Typical values for Memory Timing Registers 1 and 2 are shown in Table 2 (note that these timings were derived for a CPU bus speed of 66MHz, they may be unnecessarily conservative for 50Mhz).

Table 2. Memory Timing BCR Settings

BCR	Standard 70ns	Standard 60ns	EDO 70ns	EDO 60ns
Memory Timing Register 1	12h	11h	12h	11h
Memory Timing Register 2	8Ah	8Ah	86h	86h

- The Memory Bank Addressing Mode Registers must be set according to the type of memory installed.
- The Refresh Timer Register defaults to a value that is appropriate for a 33Mhz PCI bus, but must be changed for lower frequencies.

Table 3. Refresh Timer Register Settings

PCI Bus Frequency	Refresh Timer Register Value	Comments
33Mhz	01F8h	Default
25Mhz	0178h	
20Mhz	0128h	
16.5Mhz	00F8h	

- The Bridge Chip Set Options 3 Register bit 2 defaults to standard DRAM, and must be changed if EDO DRAM is used.

1.1.2 Software WorkArounds for Revision 1.0 Bugs

This section describes items that can be changed in software to handle errata in the V1.0 chipset. Note that software can read the Revision ID register of the '660 chipset to determine if the chipset is V1.0 (Future versions of the chipset will have Revision ID values greater than zero).

- Errata 8, Error Handling, is worked around by not enabling error handling.
- Errata 11, PCI Arbiter Parking on the 660 Bridge, is worked around by prohibiting the PCI arbiter from parking the PCI bus on the 660 Bridge. Note that the INTEL SIO defaults to this state. If an arbiter

is used that does not default to this state, then the register which controls this function must be changed. Note that this must be done while the CPU is executing code from ROM to guarantee that the register access is successful.

- Errata 18, Memory Bank Enable Register, can be worked around by modifying the software to reverse the bit order of this register from the order stated in the 660 Bridge User's Manual.
- Errata 20, Configuration Address Register vs. IDSELs, can be worked around either by using the 650 Bridge compatible method of generating type 0 PCI configuration cycles (808X XXXXh address space), or by modifying the software to adjust the device number in the Configuration Address Register according to the table described in errata 20.
- Errata 25, Memory Bank Ending Address and Memory Bank Enable BCRs, can be worked around by setting the ending address for all unused banks of memory to zero.
- Errata 30, Little Endian, can be worked around by using the address BFFF FFF7 for interrupt acknowledge while in little endian mode. The BFFF EFF0, FFFF FFF0, and FFFF FFF1 registers should not be accessed while in little endian mode.

1.1.3 601 Special Requirement

For proper L1 cache operation, the PowerPC 601 HID0 bit 31 register bit must be set to enable the HP_SNP_REQ# pin. Additionally, the 601 HP_SNP_REQ# pin must be pulled low.

1.2 Recommended Modes and Options

1.2.1 Processors

Use of the PowerPC 601, 603, 603e, and 604 processor families is supported. Workarounds are required for each CPU family.

1.2.2 Bus frequencies

The following bus frequencies (MHz) should be used (CPU internal:CPU bus:PCI bus):

- 601 and 604 . X:50:25 where X is a don't care
- 603e .... N:50:25 where N $\geq$ 75 MHz
- 603 .... 80:40:20 or 66:33:16.5

1.2.3 ROM Location

The system ROM must be direct–attached, as shown in section 8 of the 660 Bridge User's Manual. The remote ROM function does not function properly (see errata 28).

1.2.4 Strapping Options

The 660 Bridge has two strapping options, as shown in section 10.4 of the 660 Bridge User's Manual.

- CRS_C2PWXS must be pulled down to select the direct attached ROM operation.
- MWS_P2MRXS must be pulled low to disable 603 1:1 (CPU internal to CPU bus clock) mode. Note this pin must also be pulled low when a 601 or 604 is used.

The 601 has no strapping options.

The 603 has three strapping options (see section 10.7 of the *PowerPC 603 User's Manual*).

- TLBISYNC# must be pulled up to select 64–bit data bus mode.
- DRTRY# must be pulled up to select normal (not no-DRTRY#) mode.
- QACK# must be driven by HRESET# through two buffers (to delay HRESET# by at least 4ns) to select full-pinout mode.

The 604 has one strapping option (see section 8.7 of the *PowerPC 604 User's Manual*).

- DRTRY# should be pulled up to select normal (not fast–L2) mode.

1.3 Hardware WorkArounds for Revision 1.0 Bugs

The following workarounds can be implemented on boards using revision 1.0 of the 660 Bridge to fix the associated bugs. See section 3 for PLD equations files and marked up schematics that show an implementation of the required external logic.

- Both the 663 Buffer and the 664 Controller have a reserved pin that must be pulled up. This pin is not documented in the 660 Bridge User's Manual or in the reference schematic
  - 663 pin 231 must be pulled up (10k)
  - 664 pin 140 must be pulled up (10k)
- Errata 17, PCI Phase Synchronization, is worked around by providing an inverted (rather than non–inverted) PCI CLK into the PCI CLK pin of the 664 Controller. The clock skew between the inverted PCI CLK and the CPU CLK needs to +–3ns or less. This may require that the inverted PCI CLK come from a phase locked loop clock generator.
- Errata 21, ARTRY Restore, is worked around by selecting a low value for the pullup resister on ARTRY#. Recommended value is 500 ohms.
- Errata 24 and 33, Some CPU to PCI Accesses Fail During PCI Master Transactions and TLBI to PCI Space, are worked around by installing the NP2 PLD (see section 3.2), which modifies the PCI_REQ# output of the 664 Controller before it goes to the PCI arbiter and which modifies the BG# output of the 664 Controller before it goes to the CPU.
- Errata 15 and 16, 603 RWITM and 603 Snoop Bus Operations, is worked around by installing the 603TT PLD (see section 3.1), which modifies TT[1:3] between the CPU and the 664 Controller. Note that this fix is only required for operation with the 603 and 603e. This fix is not required for operation with 601 or 604 CPUs.
- Errata 29, Simultaneous Switching, is worked around either by running the CPU bus at 45MHz (50MHz works under most conditions), or by putting 33 ohm series termination resisters between the 663 Buffer CPU data bus pins and the rest of the CPU data bus. The resisters should be as close as possible to the 663.

1.4 660 Bridge Example System Implementation Schematic Set Corrections

This section lists items that are incorrect in the "Reference Implementation for the IBM27–82660 Chipset, Preliminary" schematic set, which was an addendum to the alpha level 660 Bridge User's Manual MPR660UMU–00). These items have been corrected in the "A System Implementation Example For The IBM 82660 Chipset" schematic set, which is an addendum to the beta level 660 Bridge User's Manual (MPR660UMU–01).

- Page 8, U4 pin 178 should attach to 663_CPU_PAR_ERR# (not 663_MEM_ERR# as shown).
- Page 8, U4 pin 179 should attach to 663_MEM_ERR# (not 663_CPU_PAR_ERR# as shown).
- Page 8, U3 pin 140 should be pulled high with a 10k resister.
- Page 8, U4 pin 231 should be pulled high with a 10k resister.
- Page 14, U12 pin 8 should NOT be attached to DRTRY# as shown. Disconnecting U12 pin 8 will allow DRTRY# to be pulled up at reset, signaling "normal mode" (not Fast–L2 or no–DRTRY# mode) to the CPU.
- Page 16, R18 (pullup on TAG_MATCH signal) should be 200 ohms (not 10k).

1.5 660 Bridge User's Manual Corrections

On page 13, Table 2–1 third row, change the signal name PCI_DEVIL# to PCI_DEVSEL#.

On page 72, change the reset value of the BCR from 00h to 04h.

2 Individual Errata

The following sections contain a detailed breakdown of individual 660 Bridge Errata. Errata are defined as instances in which the first silicon 660 Bridge chipset (663 Rev. 1.0 and 664 Rev. 1.0) do not perform as indicated in the beta level *660 Bridge User's Manual* (MPR660UMU–01).

Items listed with category Not Supported describe certain modes of operation that are not known to function properly. These modes are either unsimulated or are known to contain bugs which have not been fully characterized.

Items listed with category Bug describe known and characterized problems. The Details sections contain further explanation of the malfunction, and the Work Around sections describe how to avoid these bugs.

IBM plans to fix all revision 1.0 errata in revision 2.0 of the chipset. Where the planned fix requires some explanation, the section label Revision 2.0 describes the planned solution.

Errata 1 . . . . . . CPU to PCI Bus Clocks at 1:1

Date 1/26/95

Category . . . . Not Supported.

Functional . . . . . . Operation while CPU_CLK and PCI_CLK are the same frequency.

Details . . . . Bus hangs when CPU and PCI masters are accessing system memory. Snoops fail when first word is odd.

Work Around . . . Do not operate the CPU:PCI bus clocks at 1:1.

Errata 2 . . . . . . CPU to PCI Bus Clocks at 3:1

Date 1/26/95

Category . . . . Not Supported.

Functional . . . . . . Operation while the CPU_CLK is three times the frequency of PCI_CLK.

Details . . . . Unknown.

Work Around . . . Do not operate the CPU:PCI bus clocks at 3:1.

Errata 3 . . . . . . 603 in 1:1 (Internal Clock to Bus Clock) Mode

Date 1/27/95

Category . . . . Not Supported.

Functional . . . . . . Running the 603 internal clock at the same frequency as the CPU bus clock.

Details . . . . The 664 Controller fails to assert TS# on snoop cycles.

Work Around . . . Do not use 603 in 1:1 mode.

Errata 4 . . . . . . PCI Lock

Date 1/27/95

Category . . . . Not Supported.

Functional . . . . . . PCI bus master assertion of LOCK# while accessing system memory.

Details . . . .

1) In response to the initiation of a PCI to memory transaction, the 660 Bridge can begin broadcasting the associated snoop cycle on the CPU bus before knowing if the PCI transaction is a lock attempt. For PCI to memory reads, the 660 Bridge broadcasts a Clean transfer type CPU bus cycle.

   If the 660 Bridge then recognizes that the transaction is a PCI lock attempt, the Bridge is designed to generate an additional CPU bus snoop transaction of transfer type Write-with-flush. However, this re-snoop fails in CPU:PCI clock 2:1 and 3:1 configurations.

2) The snoop generated by the 660 Bridge, when a PCI bus master attempts to establish a lock, is a flush block, which fails to cancel the CPU bus reservation. A write–with–flush should be used: TT(0:4)=00010b.

Work Around . . . Do not use PCI_LOCK#.

Errata 5 . . . . . . One Clock CAS# Pulse Width

Date 1/27/95

Category . . . . Not Supported.

Functional . . . . . . Setting the memory controller CAS# pulse width to one clock.

Details . . . . Unknown

Work Around . . . Do not set CAS# pulse width to one clock (use 2, 3, or 4).

Errata 6 . . . . . . Multi-Processor Support

Date 1/27/95

Category . . . . Not Supported.

Functional . . . . . . The use of two CPU bus masters.

Details . . . . MP support fails during the following scenarios:

1) L2 cache fail. A CPU initiates a burst read, which misses in the L2. The 660 begins a memory read, and the L2 is planning to snarf (update on the fly) the memory data as it is read, but a snooping CPU ARTRY#s the CPU bus transfer. The L2 logic is improperly reset and corrupts the L2 data.

2) AACK# fail. A CPU initiates a transfer which hits in the 660 Bridge internal L2. A snooping CPU ARTRY#s the transfer. The 660 fails to assert AACK#, and the CPU bus hangs.

3) CPU_GNT# fail. Following certain CPU and PCI accesses to memory, both CPUs may be granted the CPU address bus.

4) Logic fail. A CPU initiates a CPU bus transfer. If a snooping CPU ARTRY#s that transfer and there is a previous CPU bus data tenure pending, then the state machine that tracks CPU activity hangs.

5) CPU_DBB# fail. If a CPU initiates a transfer, and a snooping CPU ARTRY#s the transfer, then the 660 Bridge may improperly terminate the data bus tenure, leaving CPU_DBB# stuck active.

Work Around . . . Do not operate the 660 Bridge with two CPU bus masters.

Errata 7 . . . . . . CPU Bus Target

Date 1/27/95

Category . . . . Not Supported.

Functional . . . . . . No external (to the 660) CPU bus targets are allowed to claim CPU bus transfers.

Details . . . .

1) If ARTRY# is asserted during a CPU bus transfer that has been claimed by a CPU bus target (using CPU_BUS_CLAIM#), the CPU bus hangs.

2) If a PCI to memory read transaction is immediately preceded by a CPU bus transfer that was claimed by a CPU bus target (using CPU_BUS_CLAIM#), a two clock MEM_RD_SMPL may occur during the PCI read, which corrupts the data returned to the PCI bus master. See Errata 12.

Work Around . . . Do not attach an external CPU bus target.

Errata 8 . . . . . . Error Handling

Date 1/27/95

Category . . . . Not Supported.

Functional . . . . . . The error detection and reporting logic is not supported.

Details . . . .

1) Reads to port BFFF EFF0 hang the bus (no AACK# or TA#), and return incorrect data.

2) A transfer type error is incorrectly detected on ROM reads that are 8-bytes or burst.

3) The value in the CPU/PCI Error Address register (index C8h) is bit reversed. Byte lanes are correct.

4) The value in the PCI Bus Error Status register (index C7h) bits 3:0 (C/BE[3:0]) is inverted.

5) The read value of the TEA mask bit in Bridge Chip Set Options 1 (index BAh) is invalid (it initially reads out as 1, whereas it is actually a 0) until it has been written .

Work Around . . . Either do not unmask the assertion of MCP# and TEA#, or do not enable error reporting for individual errors.

Errata 9 . . . . . . Worst Case Timing Parameters

Date 1/27/95

Category . . . . Not Supported.

Functional . . . . . . The chipset does not meet worst case timing requirements for 66Mhz CPU bus operation. The chipset meets worst case timing requirement for 50Mhz.

Work Around . . . Do not exceed 50Mhz CPU bus operation while operating under worst case conditions of temperature and power supply with worst case process parts.

Errata 10 .... 604+ Little Endian Alignments

Date 2/1/95

Category .... Not Supported.

Functional ...... The 604+ can generate bus operations with odd alignments and odd sizes while in little endian mode – A(29:31) = 001 or 101.

Details .... 604 bus operations in little-endian mode are unmunged incorrectly if size = 2 and CPU_ADDR[29:31] = 001 or 101, or if size = 3 and CPU_ADDR[29:31] = 000, 001, 100, or 101.

Work Around ... Do not use a 604+ in little endian mode.

Errata 11 ..... PCI Arbiter Parking on the 660 Bridge

Date 1/27/95

Category .... Bug

Functional ...... Some CPU to PCI transactions fail if PCI bus parked on 660 Bridge.

Details .... CPU to PCI transactions fail (bad read or write data) if PCI_GNT# to the Bridge is already active when the Bridge asserts PCI_REQ#. This occurs when the CPU to PCI transaction is pipelined over a memory cycle.

Work Around ... Configure the PCI arbiter such that it does not park the PCI bus on the 660 Bridge.

Errata 12 .... Two–Clock MEM_RD_SMPL on PCI master read

Date 1/26/95

Category .... Bug

Details ....

If the last access of a PCI to memory read burst is to the first word of a new cache line, then if the next operation is a PCI to memory read, its first data phase will contain corrupted data.

During the second transaction, MEM_RD_SMPL will be active for more than one clock (during the data tenure).

Work Around ... Configure the memory controller using one of the three following configurations (see the *660 Bridge User's Manual*, beta version (MPR660UMU–01) section 5.2.4):

1) For standard DRAM, set ASC = 2 clocks, or set CPW >= 4 clocks

2) For EDO DRAM, set CPW >= 4 clocks, and set ASC = 2 clocks,
or set CPW = 5 clocks,
or set CPW = 2 clocks, and set ASC = 2 clocks

Errata 13 .... Flash Lockout

Date 1/27/95

Category .... Bug

Details .... The 660 Bridge incorrectly terminates CPU writes to the flash lockout BCR, causing the CPU bus to hang.

Work Around ... Do not write to the flash lockout BCR.

Errata 14 .... 604 Fast L2 Mode

Date 2/10/95

Category .... Bug

Details ....

1) When in fast L2 mode, the 604 requires the DBG# pin to be pulsed active for exactly one clock of each data tenure. The 660 Bridge does not have s DBG# output to provide this function.

Work Around ... Don't put the 604 in fast L2 mode.

Revision 2.0 Plan A DBG# pin has been added to the revision 2.0 of the Bridge.

Errata 15 .... 603 RWITM

Date 2/14/95

Category .... Bug

Details .... To fill a cache block, the 603 generates a RWITM bus operation, even when it has no intent to modify. The 660 Bridge L2 does not snarf (update the SRAM) on RWITM operations. Coherency is not lost, but the L2 does not update on reads, so much of the benefit of the L2 is lost.

Work Around ... For operation with the 603 only, install the 603TT PLD as shown in section 3.1. This PLD transforms the 603 TT[1:3] lines to allow correct operation of the 660 Bridge L2. RWITM operations from the CPU are transformed into read operations, and other transfer types are passed thru unchanged.

Errata 16 .... 603 Broadcast Snoops

Date 2/16/95

Category .... Bug

Details .... In response to a PCI to memory transaction, the 660 Bridge generates the CPU bus operation and transfer type shown in Table 4 as Rev. 1.0 CPU Bus Snoop. However, the 603 ignores the Clean and Flush Block bus operations which the 660 Bridge broadcasts for PCI to memory reads and writes.

Table 4. Types of Snoop Cycles for PCI to Memory Operations

PCI Bus Cycle	Rev. 1.0 CPU Bus Snoop		Corrected CPU Bus Snoop For 603	
	Operation	**TT[0:4]**	**Operation**	**TT[0:4]**
Memory Read	Clean	00000	Single-Beat Read	01010
Memory Write	Flush Block	00100	Single-Beat Write with Flush	00010
Initiate Lock (Read)	Single-Beat Write with Flush	00010	Single-Beat Write with Flush	00010

Work Around ... For operation with the 603 only, install the 603TT PLD as shown in section 3.1. This PLD transforms the 660 Bridge TT[1:3] lines to support 603 snooping. The resulting snoops that the 660 Bridge broadcasts on the CPU bus are shown in Table 4 as Corrected CPU Bus Snoops For 603.

Revision 2.0 Plan The revision 2.0 chipset contains an added BCR bit (Bridge Chip Set Options 3 BCR, Index D4, bit 7) that (when set to 1) programs the 660 Bridge to interface with a 603 CPU by broadcasting the transfer codes shown in Table 4 as Corrected CPU Bus Snoops For 603.

Errata 17 .... PCI Phase Synchronization

Date 2/17/95

Category .... Bug

Details .... The PCI_CLK input logic is inverted on the 664 Controller. Consequently all PCI signals are sampled and switch off the falling (rather than rising) edge of the PCI clock. This error has only been verified during 2:1 CPU:PCI bus clock operation, but may exist in other modes as well.

Work Around ... Invert the PCI clock input to the 664 Controller. Use a fast inverter to minimize clock skew (in most instances a delay of 3ns is acceptable).

Errata 18 .... Memory Bank Enable Register

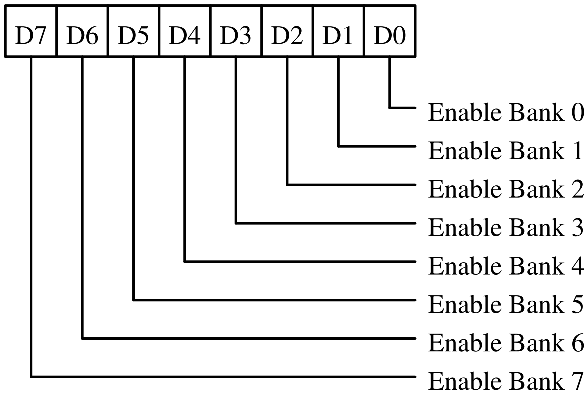
Date 2/17/95

Category .... Bug

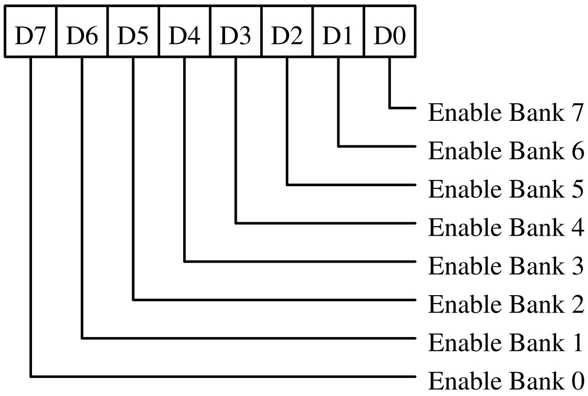
Details ....

1) The bits are endian–reversed. The correct order is bit 0 enables bank 0, bit 1 enables bank 1,..., bit 7 enables bank 7. The actual (incorrect) order is bit 0 enables bank 7, bit 1 enables bank 6, etc..

Correct Bit to Bank Mapping



Actual (Incorrect) Bit to Bank Mapping



2) The reset value of this register is incorrect. The correct reset value is 01h. The reset value in the *User's Manual* section 4.3.26 is incorrectly shown as 00h. The actual (incorrect) reset value is 08h. This is the actual value of the read data, uncorrected for bit to bank mapping. This BCR is (incorrectly) set to enable bank 4 coming out of reset.

Work Around ... Adjust the memory controller initialization microcode to adapt to the actual bit ordering and reset value. Set this register to enable only the populated memory banks.

Errata 19 .... 650 Bridge Compatible PCI Configuration

Date 2/17/95

Category .... Bug

Details ....

The 660 Bridge allows the CPU to initiate type 0 PCI configuration transactions using two different software methods. A typical hardware implementation of the type 0 configuration transaction is to resistively couple certain PCI_AD lines to IDSEL lines, as described in the *PCI Local Bus Specification Revision 2.0* in section 3.6.4. To support this implementation, the 660 Bridge is designed to allow the IDSEL lines time to become valid and stable by enabling its PCI_AD drivers one PCI_CLK before asserting FRAME#. When software uses the CFC/CF8 indexed BCR method of generating type 0 PCI configuration transactions, the hardware operates correctly.

However, when software uses the 650 Bridge compatible method of generating type 0 PCI configuration transactions (by initiating memory accesses to 8080-series locations as described in the *User's Manual* sections 4.2 and 6.3.4), the 660 Bridge fails to pre-drive the PCI_AD lines. This may not allow the IDSEL lines enough time to become settle into a valid state.

Work Around ... If there is a timing problem with resistively coupled IDSELs, consider changing the coupling resistor values to 0 ohms. It is also be possible to modify the software to use the CFC/CF8 method of generating these transactions.

Errata 20 .... Configuration Address Register vs. IDSELs

Date 2/17/95

Category .... Bug

Details .... When generating PCI type 0 configuration cycles using the CF8/CFC registers, the value that software writes to the to the Device Number field of the Configuration Address Register is incorrectly decoded as shown in Table 5. Note that device number 0, the internal decode, correctly decodes to the internal BCRs.

Table 5. CFC/CF8 IDSEL Generation

Device Number	Correct PCI_AD and IDSEL Assertion		Actual PCI_AD and IDSEL Assertion	
	PCI_AD	IDSEL	PCI_AD	IDSEL
0	Internal	Internal	Internal	Internal
1	11	0	25	14
2	12	1	24	13
3	13	2	23	12
4	14	3	22	11
5	15	4	21	10
6	16	5	20	9
7	17	6	19	8
8	18	7	18	7
9	19	8	17	6
10	20	9	16	5
11	21	10	15	4
12	22	11	14	3
13	23	12	13	2
14	24	13	12	1
15	25	14	11	0
16	26	15	none	none
17	27	16	none	none
18	28	17	none	none
19	29	18	none	none
20	30	19	none	none
21	31	20	none	none
22–31	n/a	n/a	n/a	n/a

Work Around ...

Modify the PCI configuration software to modify the device number field such that the desired IDSEL is activated. Thus to activate IDSEL 0, change the device number from 1 to 15. To activate IDSEL 10, change the device number from 11 to 5. ... To activate IDSEL 14, change the device number from 15 to 1. Note that IDSELs 16:21 are not available.

Alternately, the physical IDSEL lines can be coupled to different PCI_AD lines such that IDSEL 0 is coupled to PCI_AD[25], IDSEL 1 is coupled to PCI_AD[24], ..., and IDSEL 14 is coupled to PCI_AD[11]. Note that IDSELs 16:21 are not available.

Errata 21 .... ARTRY# restore

Date 2/17/95

Category .... Bug

Details .... When the 660 Bridge asserts ARTRY#, it does not restore it to the deasserted state before tristating it. Instead, ARTRY# is merely tristated.

Work Around ... Replace the pullup resister on ARTRY# with one of a value low enough to pull ARTRY# high in three CPU CLK cycles (330 to 500 ohm range).

Errata 22 .... Memory Controller Misc BCR (8000 0821h)

Date 2/22/95

Category .... Bug

Details ....

When the CPU writes this register, the data is latched into the BCR from CPU byte lane 0, instead of the correct byte lane 1.

The 601, 603, and 604 CPUs mirror 1-byte write data on all 8 data byte lanes, so correct data is written to the register. However, the ESP does not mirror the data on all data byte lanes, so the write data is incorrect.

Work Around ... Don't write this register from the ESP (RISC WATCH).

Errata 23 .... CPU Address Bus Pre–drive During Snoops

Date 2/22/95

Category .... Bug

Details ....

When a PCI bus master accesses memory, the 660 Bridge is designed to broadcast a snoop cycle on the CPU bus to support cache coherency. During the snoop cycle, CPU_ADDR[0:31] and TT[0:4] are to be asserted one CPU_CLK before TS# is asserted, in order to allow these lines plenty of time to settle before the CPU(s) sample TS# active.

However, the 660 Bridge fails to pre-drive these lines, and asserts CPU_ADDR[0:31] and TT[0:4] on the same CPU_CLK that it asserts TS#.

Work Around ... If any of CPU_ADDR[0:31] is a timing-critical path, this may limit the maximum operating frequency of the system. The CPU_CLK frequency can be reduced to allow sufficient time for the CPU_ADDR lines to settle.

Errata 24 .... Some CPU to PCI Accesses Fail During PCI Master Transactions

Date 3/15/95

Category .... Bug

Details ....

1) CPU to PCI accesses fail following a collision between a CPU to PCI transaction and a PCI to memory transaction.

The bug is encountered when the following sequence of events occur: a CPU to PCI access begins, and the 660 Bridge asserts PCI_REQ# to the PCI arbiter. But before the Bridge receives PCI_GNT#, a PCI bus master initiates a PCI to memory transaction. In order to broadcast the address of the PCI to memory transaction as a snoop cycle on the CPU bus, the 660 Bridge correctly asserts ARTRY# to back off the CPU.

However, the 660 Bridge incorrectly leaves PCI_REQ# asserted. Since PCI_REQ# is active, the PCI arbiter can assert PCI_GNT# to the Bridge before the Bridge allows another CPU to PCI access. If this occurs, then the bug described in Errata 11 is encountered (CPU to PCI access fails if PCI bus is parked on the Bridge).

2) CPU to PCI accesses get duplicated on the PCI bus. As a result of the CPU initiating a single CPU to PCI transaction, two PCI transactions are generated.

This bug is encountered only when the 660 Bridge retries a PCI to memory burst by asserting STOP such that it is active on the same clock that the Bridge samples TS# asserted at the beginning of a CPU to PCI transaction, and only if a CPU to memory data tenure is outstanding.

In this situation, the 660 Bridge will have retried the PCI bus master only as the result of a CPU having retried the snoop that the 660 Bridge broadcast on the CPU bus due to the PCI to memory transaction. For a TS# to then be asserted while the PCI cycle is being STOPed, the broadcast snoop must have been retried because the CPU is currently pushing the snooped cache line. For a memory data tenure (the snoop push operation) to be outstanding, the memory cycle must be very slow, and this only occurs when refresh hits during this cycle.

The result is that the internal state of the 660 Bridge is set such that the Bridge will ARTRY# the CPU to PCI transaction, which causes the CPU to rerun that transfer. However, the bridge still generates the PCI part of the CPU to PCI transaction. This results in executing the transaction twice on the PCI.

3) CPU to PCI writes get ARTRY#'d and posted. If a CPU has initiated a CPU to PCI write that the 660 Bridge has not yet posted, and a PCI master initiates a PCI to memory transaction, then the CPU to PCI write is ARTRY#'d, but the 660 Bridge write posting logic thinks that the posting occurred.

The result is that the incorrectly posted cycle executes on the PCI bus. The address and C/BE information is incorrect.

Work Around ...

1) The first bug can be avoided by masking PCI_REQ# from the 664 Controller between the conflict and the next CPU to PCI access.

2) The second bug can be avoided by masking BG# to the CPU between a snoop hit while a data tenure is outstanding and the end of the data tenure.

3) The third bug can be avoided by only allowing BG# to be asserted to the CPU when PCI_CLK is low.

These bugs are corrected by the NP2 PLD described in section 3.2.

Errata 25 .... Memory Bank Ending Address and Memory Bank Enable BCRs

Date 3/06/95

Category .... Bug

Details ....

If the Memory Bank Ending Address BCRs for disabled banks of memory are non–zero, then PCI accesses to memory may fail.

The address of the top of memory is calculated from the Memory Bank Ending Address BCRs, essentially without regard to the bank enables. If the Memory Bank Ending Address BCR pair for a disabled bank contains a value larger than the true top of memory, then the top of memory address will be calculated incorrectly. As a (non-obvious) result, the 660 Bridge may not assert DEVSEL# during PCI to memory transactions, causing the fail.

Work Around ... Memory controller setup software should set the Memory Bank Ending Address BCRs (index 90h–97h) and the Memory Bank Extended Ending Address BCRs (index 98h–9Fh) to zero for all disabled banks of memory.

Errata 26 .... Asynchronous L2 Address Hold Time

Date 4/18/95

Category .... Bug

Details ....

The address bus into the data SRAMs is not held valid until the rising edge of WE#, as is required by the SRAMs.

1) The SRAM_ALE signal goes high on the same clock as SRAM_WE#. Due to heavier loading on SRAM_WE#, the address to the SRAM may change before SRAM_WE# rises.

2) SRAM_A0 and SRAM_A1 switch on the same clock as SRAM_WE#. Due to heavier loading on SRAM_WE#, these signals may may switch before SRAM_WE# rises.

3) The first data beat of a pipelined burst write in which the L2 is updating will end (TA#) too soon to meet the write data hold time requirement (data hold time from SRAM_WE# rising).

Work Around ...

1) & 2) Delay SRAM_ALE, SRAM_A0 (SRAM_ADS), and SRAM_A1 (SRAM_CNT) enough to ensure appropriate address hold time. In most cases an F244 type buffer will provide sufficient delay.

3) Delay TA# between the 664 Controller and the CPU by one CPU clock on burst write cycles.

Errata 27 .... PCI Disconnect Timer

Date 4/18/95

Category .... Bug

Details .... PCI disconnect timer does not work. This does not appear to cause any fails per se, but the timer never times out.

Work Around ... Don’t enable the PCI disconnect timer.

Errata 28 .... Remote ROM

Date 4/18/95

Category .... Bug

Details ....

1) Remote ROM accesses of 8 bytes or burst sizes fail. They are incorrectly detected as having unsupported transfer types.

2) Remote ROM accesses following a write to the PCI return bad data on byte lane 0. Therefore remote ROM reads that use byte lane 0 fail.

Work Around ... Use direct–attach ROM.

Errata 29 .... ECIWX and EXOWX

Date 4/18/95

Category .... Bug

Details .... The ECIWX and EXOWX bus operations cause memory accesses which can corrupt memory.

Work Around ... Don't execute ECIWX and EXOWX instructions.

Errata 30 .... Simultaneous Switching

Date 4/18/95

Category .... Bug

Details .... Simultaneous switching noise generated when the 663 Buffer switches a large number of signals on the CPU data bus causes the CPU to read incorrect data at frequencies greater than or equal to 50Mhz.

Work Around ...

Put series termination 33 ohm resisters on 663 Buffer pins CPU_DATA[0:63], as close to the 663 pins as possible.

OR run CPU_CLK below 50Mhz. (45Mhz should be safe).

Errata 31 .... Little Endian

Date 4/18/95

Category .... Bug

Details .... When in little endian mode, the 660 Bridge fails to correctly decode CPU transfers to the following addresses:

BFFF EFF0 – System Error Address Register
BFFF FFF0 – Interrupt Acknowledge
FFFF FFF0 – ROM Write Control Register
FFFF FFF1 – ROM Lockout Register

Work Around ...

Don't use BFFF EFF0, FFFF FFF0, and FFFF FFF1 registers in little endian mode.

Interrupt acknowledge in little endian mode can be accessed at address BFFF FFF7.

Errata 32 .... Special Port 92 Mirror

Date 4/18/95

Category .... Bug

Details ....

When writing this register, the data is latched from CPU byte lane 0, instead of the correct CPU byte lane 2.

When executing code from the CPU, write data is mirrored on all the byte lanes, thus the correct data is written. However, the ESP does not mirror the byte lanes, thus the write data is random.

Work Around ... Don't write this register from the ESP (RISC WATCH).

Errata 33 .... TLBI to PCI Space

Date 4/24/95

Category .... Bug

Details .... If the CPU runs a TLB Invalidate bus operation with an address that is in PCI space then PCI_REQ# goes active and hangs active until (if) the CPU runs a normal cycle to PCI space.

Work Around ... Mask PCI_REQ# from time of TLB invalidate until a normal normal cycle to PCI space occurs. This workaround is incorporated into the NP2 PLD that is required by errata #24.

3 Implementation Details

3.1 603 Transfer Type PLD

3.1.1 Design File

```
;---------------------------------------------------------------------
;
;    File Name:    603TT.PDS
;    Fixes Errata:  #15 & #16
;    Basic Operation:  Munges TT(1:3) between CPU & '664 Controller
;    PAL Type: PAL16V8-5, 20-pin
;    PAL Equation Format: PALASM
;
;
;-------------------------- Declaration Segment -------------------
TITLE    603TT.pds
PATTERN  none
REVISION 1.0
AUTHOR   Ben Drerup
COMPANY  IBM
DATE     03/15/95

CHIP  _603tt  PALCE16V8

;--------------------------- PIN Declarations ---------------------
;
;  Inputs
;
PIN  1    CPU_CLK                  ; CLOCK

PIN  2    CPU_BG_                  ; INPUT
PIN  3    CPU_ABB_                 ; INPUT

;
;  Bi-directional
;
PIN 13    664_TT1         COMB     ; COMBINATIONAL
PIN 14    664_TT2         COMB     ; COMBINATIONAL
PIN 15    664_TT3         COMB     ; COMBINATIONAL
PIN 16    CPU_TT1         COMB     ; COMBINATIONAL
PIN 17    CPU_TT2         COMB     ; COMBINATIONAL
PIN 18    CPU_TT3         COMB     ; COMBINATIONAL

;
;
;  Internal Registers
;
PIN 12    LAT_CPU_BG_     REG      ; REGISTERED
PIN 19    LAT_CPU_ABB_    REG      ; REGISTERED

;
;------------------ Boolean Equation Segment ---------------------
```

```
;
EQUATIONS
;
; Output enables for TT bits (Reg OE's controlled by pin 11)
;
664_TT1.TRST = /CPU_ABB_       ; This equation results in long timing
664_TT2.TRST = /CPU_ABB_       ; path for TT bits from CPU to 664
664_TT3.TRST = /CPU_ABB_

CPU_TT1.TRST = CPU_BG_ * LAT_CPU_BG_ * CPU_ABB_ * LAT_CPU_ABB_
CPU_TT2.TRST = CPU_BG_ * LAT_CPU_BG_ * CPU_ABB_ * LAT_CPU_ABB_
CPU_TT3.TRST = CPU_BG_ * LAT_CPU_BG_ * CPU_ABB_ * LAT_CPU_ABB_

;
; Equations for Internal Registers
;

LAT_CPU_BG_  = CPU_BG_

LAT_CPU_ABB_ = CPU_ABB_

;
; Combinational equations for TT bits.
;

664_TT1 = CPU_TT1

664_TT2 = CPU_TT2 * /CPU_TT1
         +CPU_TT2 * /CPU_TT3

664_TT3 = CPU_TT3

CPU_TT1 = /664_TT2

CPU_TT2 = GND

CPU_TT3 = VCC

;---------------------------- END OF FILE --------------------------
```