intel®

i960® Rx I/O PROCESSOR SPECIFICATION UPDATE

Release Date: March, 1997
Order Number: 272918-007

The i960® Rx I/O Processor may contain design defects or errors known as errata. Characterized errata that may cause the i960® Rx I/O Processor's behavior to deviate from published specifications are documented in this specification update.

REVISION HISTORY

Rev. Date	Version	Description
3/12/97	007	Added specification clarification 8. In SUMMARY TABLE OF CHANGES Errata table item 10, added “X” for 80960RP 33/3.3 and the 80960RD 66/3.3 A0 step.
2/10/97	006	Added errata items 21, 22. Added specification change items 7, 8, 9, 10, 11 (the previous specification change item #6 is now specification clarification item 5). Added specification clarification items 6, 7. Changed the status of errata items 15, 19, 20. Changed the stepping information for specification change item 4. Modified documentation change item 39. Added new stepping information; see IDENTIFICATION INFORMATION. References to “80960RP” have changed to 80960RP 33/3.3, 80960RP 33/5.0, 80960RD 66/3.3, where appropriate, to reflect the new steppings. See SUMMARY TABLE OF CHANGES. In the ERRATA descriptions, the “Status” heading has been removed. Refer to the Errata table in SUMMARY TABLE OF CHANGES for status.
1/15/97	005	Added errata items 18, 19. Added three documentation changes. See Summary Table of Changes for complete list.
12/04/96	004	Added errata items 14, 15, 16, 17. Added specification clarification item 5. Added several documentation changes. See Summary Table of Changes for complete list.
11/01/96	003	Added errata items 12, 13. Changed the status of errata item 2 from Eval to Fix. Added Specification Change item 5. Added several documentation changes. See Summary Table of Changes for complete list. Changed numbering sequences for Revision History, Errata, Specification Changes, Specification Clarifications and Document Changes. Added subheadings under Document Changes to separate individual document types (i.e., User’s Manual, Addendum, etc.).
9/6/96	002	Added errata items 9, 10, 11. Added information for the A-1 stepping. Added Specification Change 4.
7/15/96	001	This is the new Specification Update document. It contains all identified errata published prior to this date.

PREFACE

As of July, 1996, Intel has consolidated available historical device and documentation errata into this new document type called the Specification Update. We have endeavored to include all documented errata in the consolidation process, however, we make no representations or warranties concerning the completeness of the Specification Update.

This document is an update to the specifications contained in the Affected Documents/Related Documents table below. This document is a compilation of device and documentation errata, specification clarifications and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Information types defined in Nomenclature are consolidated into the specification update and are no longer published in other documents.

This document may also contain information that was not previously published.

Affected Documents/Related Documents

Title	Order
i960® RP Microprocessor User's Manual	272736-001
i960® RP I/O Processor at 5 Volts Data Sheet	272737-003
i960® RP/RD I/O Processor at 3.3 Volts Data Sheet	273001-001

Nomenclature

Errata are design defects or errors. These may cause the published (component, board, system) behavior to deviate from published specifications. Hardware and software designed to be used with any component, board, and system must consider all errata documented.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in any new release of the specification.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

NOTE:

Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

SUMMARY TABLE OF CHANGES

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the 80960Rx product. Intel may fix some of the errata in a future stepping of the component, and account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Errata exists in the stepping indicated. Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Page

(Page):	Page location of item in this document.

Status

Doc:	Document change or update will be implemented.
Fix:	This erratum is intended to be fixed in a future step of the component.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
Eval:	Plans to fix this erratum are under evaluation.

Row

┃	Change bar to left of table row indicates this erratum is either new or modified from the previous version of the document.

Errata (Sheet 1 of 2)

Item	Stepping			Page	Status	ERRATA
	80960RP 33/5.0		80960 RP/RD 33/66/3.3			
	A-0	A-1	A-0			
1	X	X		13	Fix	Loss of local grant signal during an expansion ROM access
2	X	X		13	Fix	Byte enables, BE1:0#, can cause hold violations to flash memory
3	X	X		14	Fix	The loss of HOLD on the 80960 local bus from an external bus master during a DRAM refresh cycle could cause bus ownership issues with the primary ATU, the Messaging Unit, or the expansion ROM
4	X	X		14	Fix	Unaligned DMA read transfers could prevent further operation of DMA Unit
5	X	X		15	Fix	Bridge Control Register can only be written as a 16-bit value
6	X	X	X	15	NoFix	Parity checking for inbound PCI address cycles is always enabled for the ATU
7	X	X		16	Fix	Configuration Write Cycle occurring simulta-neously with clearing the Configuration Cycle Retry bit causes a deadlock problem
8	X	X		16	Fix	PCI Master Parity Error bits set in Bridge Interrupt Status Registers regardless of whether parity checking is enabled
9	X			17	Fix	Deadlock condition under simultaneous inbound read and outbound write PCI traffic through secondary ATU
10	X		X	17	Fix	Intermittent ONCE Mode upon power-up
11	X	X		18	Fix	ATU lockup condition under simultaneous outbound read data and inbound write data through the inbound data queue
12	X	X		18	Fix	PCI-to-PCI bridge can corrupt data during Memory Write and Invalidate cycles that insert IRDY# wait states during data-data transfers
13	X	X		19	Fix	Using the Messaging Unit circular queues with more than one recovery wait state on the local bus can cause primary Address Translation Unit data corruption

Errata (Sheet 2 of 2)

Item	Stepping			Page	Status	ERRATA
	80960RP 33/5.0		80960 RP/RD 33/66/3.3			
	A-0	A-1	A-0			
14	X	X		20	Fix	Access to non-existent DRAM causes incorrect error reporting
15	X	X	X	20	NoFix	Address Translation Unit write and Messaging Unit queue port write may get out of order on the 80960 local bus
16	X	X		21	Fix	Using multiple DMA channels that perform unaligned transfers can cause a DMA channel performing a read to stop operating
17	X	X		22	Fix	Messaging Unit logs multiple interrupt sources when using the index registers
18	X	X	X	24	Eval	DMA Descriptors appended to the end of a chain may not execute
19	X	X	X	23	NoFix	Downstream exclusive read transaction through multiple bridges may cause a deadlock condition
20	X	X	X	23	Fix	Inbound ATU PCI read issue with retried completion cycles
21	X	X	X	24	Fix	Null writes (BE3:0# deasserted) to the MU Inbound and Outbound Queue Ports corrupt the corresponding queue pointer
22	X	X		24	Fix	PCI-to-PCI Bridge Unit can corrupt data during Memory Write cycles when wait states are inserted simultaneously on both the primary and secondary sides of the bridge

Specification Changes

Item	Stepping			Page	Status	SPECIFICATION CHANGES
	80960RP 33/5.0		80960 RP/RD 33/66/ 3.3			
	A-0	A-1	A-0			
1	X	X		25	Doc	PCI Interrupt Routing Select Register (PIRSR) Polarity
2	X	X	X	25	Doc	Asynchronous clocking mode not supported
3	X	X	X	25	Doc	The MU interlock mechanism remains enabled for APIC registers when the APIC is disabled
4		X	X	26	Doc	Incorrect BIOS access of block size from Base Address Registers
5	X	X	X	26	Doc	ATU Configuration Register Bit 12 Definition is Changed
6				27	Doc	Multiple reads of the Base Address Register after writing all 1's will return different values (See Specification Clarification Item #5)
7			X	27	Doc	Additional devices can be configured as public or private on the secondary PCI bus by programming the Secondary IDSEL Select Register (SISR)
8			X	27	Doc	PCI Interrupt Routing Select Register (PIRSR) now supports individual routing of each XINT3:0# pin
9			X	27	Doc	The Memory Bank Extended MWE3:0# bits in the Memory Bank Control Register can provide one clock of address hold time during write cycles
10			X	28	Doc	Mask bits added for all PCI-PCI bridge error conditions which may cause an NMI# to the i960 core processor
11			X	29	Doc	Multiple reads of the Base Address Register, after writing all 1's, returns the limit register value until rewritten

Specification Clarifications

Item	Stepping			Page	Status	SPECIFICATION CLARIFICATIONS
	80960RP 33/5.0		80960 RP/RD 33/66/ 3.3			
	A-0	A-1	A-0			
1	X	X		30	Doc	Five IDSEL lines for the secondary PCI bus are not enabled
2	X	X	X	30	Doc	How to use the Data Enable (DEN#) Signal with an In-Circuit Emulator
3	X	X	X	31	Doc	Accessing 80960Rx MMRs via the primary PCI bus
4	X	X	X	32	Doc	Parity on data cycles not checked before delayed write completion cycles are accepted
5	X	X		32	Doc	Multiple reads of the Base Address Register after writing all 1's will return different values
6	X	X	X	32	Doc	When determining memory address block size, accesses to the Base Address Register must be 32-bit configuration cycles
7	X	X	X	33	Doc	Secondary PCI bus reset notification to i960 core processor
8	X	X	X	34	Doc	Parity error reporting during configuration cycles

Documentation Changes (Sheet 1 of 4)

Item	Document Revision	Page	DOCUMENTATION CHANGES
1.	272736-001	35	Section 8.2.2, Page 8-16
2.	272736-001	35	Section 8.3.3, Page 8-27
3.	272736-001	35	Section 8.4.7, Page 8-36
4.	272736-001	36	Section 8.4.7, Page 8-37, Table 8-10
5.	272736-001	36	Section 9.10.8, Page 9-34
6.	272736-001	36	Section 11.2.1, Page 11-3
7.	272736-001	36	Section 11.2.8, Page 11-6
8.	272736-001	37	Section 11.4, Page 11-12
9.	272736-001	37	Section 11.6, Page 11-24
10.	272736-001	37	Section 11.6.1, Page 11-24

Documentation Changes (Sheet 2 of 4)

Item	Document Revision	Page	DOCUMENTATION CHANGES
11.	272736-001	37	Section 13.3.7, Page 13-18
12.	272736-001	37	Section 13.6, Page 13-24
13.	272736-001	37	Section 13.6.1, Page 13-26
14.	272736-001	38	Section 14.4.1, Page 14-8, Table 14-2
15.	272736-001	38	Section 14.4.5, Page 14-14, Figure 14-5
16.	272736-001	38	Section 14.4.5, Page 14-15, Figure 14-6
17.	272736-001	38	Section 14.4.5, Page 14-16, Figure 14-7
18.	272736-001	39	Section 14.4.5, Page 14-16, Figure 14-8
19.	272736-001	39	Section 14.4.5, Page 14-17, Figure 14-9
20.	272736-001	39	Section 14.5, Page 14-18, Table 14-6
21.	272736-001	39	Section 14.5.1, Page 14-22, Table 14-8
22.	272736-001	39	Section 14.5.4, Page 14-24, Table 14-9
23.	272736-001	40	Section 14.5.6, Page 14-27, Table 14-11
24.	272736-001	40	Section 14.5.7, Page 14-29, Table 14-12
25.	272736-001	41	Section 14.5.8, Page 14-31, Table 14-13
26.	272736-001	41	Section 14.6.2, Page 14-33, Table 14-15
27.	272736-001	41	Section 14.7.2, Page 14-38 and 39, Figures 14-17 and 18
28.	272736-001	41	Section 15.5.2.1, Page 15-12
29.	272736-001	41	Section 15.5.6, Page 15-16
30.	272736-001	41	Section 15.5.6, Page 15-19 and 15-20, Table 15-6
31.	272736-001	41	Section 15.6.2.1, Page 15-22
32.	272736-001	41	Section 15.6.2.2, Page 15-23
33.	272736-001	41	Section 15.12.5, Page 15-37
34.	272736-001	42	Section 15.13, Page 15-37
35.	272736-001	42	Section 15.13, Page 15-38, Figure 15-8
36.	272736-001	42	Section 15.13.4, Page 15-42, Table 15-16
37.	272736-001	43	Section 15.13.7, Page 15-44, Table 15-19
38.	272736-001	43	Section 15.13.16, Page 15-50, Table 15-28
39.	272736-001	43	Section 15.13.23, Page 15-54, Table 15-35
40.	272736-001	48	Section 15.13.23, Pages 15-54, Table 15-34
41.	272736-001	48	Section 16.1, Page 16-2

Documentation Changes (Sheet 3 of 4)

Item	Document Revision	Page	DOCUMENTATION CHANGES
42.	272736-001	48	Section 16.2.1, Page 16-5, Figure 16-2
43.	272736-001	48	Section 16.2.1, Page 16-5
44.	272736-001	49	Section 16.2.2, Page 16-7
45.	272736-001	49	Section 16.2.2, Page 16-8
46.	272736-001	49	Section 16.2.3, Page 16-8
47.	272736-001	49	Section 16.2.6.1, Page 16-10
48.	272736-001	50	Section 16.2.6.1, Page 16-11, Figure 16-4
49.	272736-001	50	Section 16.2.6.1, Page 16-12
50.	272736-001	50	Section 16.2.6.1, Page 16-13, Figure 16-5
51.	272736-001	50	Section 16.2.6.2, Page 16-14
52.	272736-001	50	Section 16.5, Page 16-18, Figure 16-7
53.	272736-001	50	Section 16.5.1, Page 16-18
54.	272736-001	50	Section 16.6, Page 16-20, Table 16-3
55.	272736-001	51	Section 16.7, Page 16-24
56.	272736-001	51	Section 16.7, Page 16-25, Figure 16-8
57.	272736-001	51	Section 16.7, Page 16-26, Table 16-8
58.	272736-001	51	Section 16.7.2, Page 16-29, Table 16-11
59.	272736-001	51	Section 16.7.4, page 16-31
60.	272736-001	51	Section 16.7.7, Page 16-33, Table 16-16
61.	272736-001	51	Section 16.7.8, Page 16-33, Table 16-17
62.	272736-001	51	Section 16.7.9, Page 16-34, Table 16-18
63.	272736-001	51	Section 16.7.11, Page 16-35
64.	272736-001	52	Section 16.7.14.1, Page 16-38
65.	272736-001	52	Section 16.7.19, Page 16-43, Table 16-30
66.	272736-001	52	Section 16.7.21, Page 16-44
67.	272736-001	52	Section 16.7.30, Page 16-50
68.	272736-001	53	Section 16.7.32, Page 16-51, Table 16-43
69.	272736-001	56	Section 16.7.33, page 16-53, Table 16-44
70.	272736-001	56	Section 16.7.33, Page 16-54, Table 16-44
71.	272736-001	56	Section 16.7.34, page 16-54, Table 16-45
72.	272736-001	56	Section 16.7.34, Page 16-55, Table 16-45

Documentation Changes (Sheet 4 of 4)

Item	Document Revision	Page	DOCUMENTATION CHANGES
73.	272736-001	56	Section 16.7.35, Page 16-56, Table 16-46
74.	272736-001	56	Section 16.7.36, page 16-57
75.	272736-001	57	Section 16.7.37, page 16-57, Table 16-47
76.	272736-001	57	Section 16.7.39, Page 16-58
77.	272736-001	57	Section 16.7.40, Page 16-59
78.	272736-001	57	Section 16.7.41, Page 16-59 and 60
79.	272736-001	58	Section 16.7.41.1, Page 16-39
80.	272736-001	58	Section 16.7.42, Page 16-60
81.	272736-001	58	Section 17.7.20, Page 17-30, Figure 17-3
82.	272736-001	58	Section 18.2.3, Page 18-6
83.	272736-001	59	Section 18.3.2, Page 18-11
84.	272736-001	59	Section 18.3.2, Page 18-11, Table 18-9
85.	272736-001	59	Section 20.4, Page 20-11
86.	272736-001	59	Section 20.4, Page 20-17, Figure 20-12
87.	272736-001	60	Section 20.4, Page 20-18, Figure 20-13
88.	272736-001	60	Section 20.5.1, Page 20-19
89.	272736-001	60	Section 20.5.2, Page 20-20
90.	272736-001	60	Section 22.2, Page 22-1
91.	272736-001	60	Section 23.2.4.4, Page 23-9, Table 23-4
92.	272736-001	63	Appendix A, Page A-4, Table A-5

IDENTIFICATION INFORMATION

Markings

80960Rx processors may be identified electrically according to device type and stepping. Refer to the *i960® RP I/O Processor at 5 Volts* Data Sheet or the *i960® RP/RD I/O Processor at 3.3 Volts* Data Sheet for instructions on how to obtain the identifier number.

Device and Stepping	Processor Device ID Register (PDIDR) (g0)	PCI-to-PCI Bridge Unit Revision ID Register (RIDR) (1008H)	Address Translation Unit Revision ID Register (ATURID) (1208H)
80960RP 33/5.0 A-0	0x00860013	0x00	0x00
80960RP 33/5.0 A-1	0x10860013	0x01	0x01
80960RP 33/3.3 A-0 80960RD 66/3.3 A-0	0x08861013	0x02	0x02

ERRATA

1. Loss of local grant signal during an expansion ROM access

PROBLEM: This occurs when the memory controller wait state MMRs are set to their maximum number of wait states. On an primary PCI inbound access through the expansion ROM window, ATU REQ# is deasserted during an 80960 local bus recovery state. This causes GNT# to deassert and arbitration is granted to the i960 core processor, but the ATU is still mastering recovery cycles. Contention occurs on the bus, the i960 core processor hangs, and incorrect data is returned from the expansion ROM access.

IMPLICATION: Under the conditions outlined above, when the workaround is not used, this erratum could cause: internal bus contention, and erroneous data to be returned from the expansion ROM access.

WORKAROUND: Do not allow other masters (ATUs, DMAs) to access the 80960 local bus while reads to the expansion ROM window are occurring. In a typical system, expansion ROM accesses only occur during the initialization sequence, while DMAs and ATUs are typically inactive. In addition, program the memory controller programmable recovery states for ROM accesses (T_{WRR}) to 0 or 1.

2. Byte enables, BE1:0#, can cause hold violations to flash memory

PROBLEM: When communicating with 8-bit memory, the Memory Controller Unit (MCU) provides address bits 13:2 on pins MA11:0, and address bits 1:0 on pins BE1:0#. The address is not held on the BE1:0# pins during recovery cycles. Some flash memories specify an address hold time relative to the deassertion of WE#. This is not a problem with the MA pins, but is a problem with the BE1:0# pins when they act as address signals.

IMPLICATION: Under the conditions outlined above, this erratum could cause problems with programming on-board flash devices if one of the workarounds is not used. This erratum only occurs during writes to flash memory. Read operations are not affected. In particular, the 80960Rx will not encounter any problems reading from a “pre-programmed” flash device.

WORKAROUND: Adhere to the following:

1. Use flash devices that have a zero (0 ns) address hold time requirement.
2. Externally buffer BE1:0# to provide the necessary hold time.

3. The loss of HOLD on the 80960 local bus from an external bus master during a DRAM refresh cycle could cause bus ownership issues with the primary ATU, the Messaging Unit, or the expansion ROM

PROBLEM: The following events must happen for this erratum to occur:

1. A normal inbound read on the primary or secondary ATU must be occurring.
2. An MU read or an expansion ROM read is requesting the bus.
3. An external bus master requests the bus by asserting the HOLD pin.
4. A DRAM refresh occurs while the ATU is still the bus master. During the refresh, the HOLD pin is deasserted. After the refresh, both the ATU (to finish its read transaction) and the MU/expansion ROM interface simultaneously own the bus. This causes bus contention.

IMPLICATION: Under the conditions outlined above, this erratum could cause internal bus contention if the workaround is not used.

WORKAROUND: Do not allow an external bus master that is requesting the 80960 local bus to deassert the HOLD pin without first receiving HOLDA.

4. Unaligned DMA read transfers could prevent further operation of DMA Unit

PROBLEM: This problem occurs with unaligned DMA transfers that use memory read, memory read line, or memory read multiple PCI commands. During an unaligned DMA Read transfer, the DMA “builds” complete WORDs in its buffer from the unaligned data being read from the PCI bus. If the DMA unit loses control of the 80960 local bus after two consecutive local bus cycles AND there is only a partial WORD (1, 2 or 3 bytes) remaining in the DMA buffer to be written into 80960Rx local memory, that partial WORD is never transferred. The partial WORD remains in the DMA buffer and prevents further operation of the DMA channel.

Conditions or modes which increase the likelihood of losing control of the 80960 local bus on consecutive cycles include: demand mode DMA, DMA transfers that cross a 2 Kbyte boundary, having the Local Bus Arbitration Latency Counter Register (LBALCR) set to a low value.

IMPLICATION: Under the conditions outlined above, if one of the workarounds is not followed, the DMA channel could cease to function.

WORKAROUND: Either of two workarounds can prevent this erratum:

1. In demand mode, perform only aligned transfers.

2. In block transfer mode, make sure that unaligned DMA transfers do not cross and then end on a 2 Kbyte boundary with a partial word to transfer. This can be done by adding the destination address register and the byte count register and logically ANDing that result with 0x000007FF. If the final result is either 1, 2 or 3 bytes, then the DMA transfer should be broken up into two separate transfers where the final transfer moves the partial word. This can easily be accomplished using the DMA chaining mode on the 80960Rx. The LBALCR register should also be programmed to a value greater than 10.

5. Bridge Control Register can only be written as a 16-bit value

PROBLEM: The 80960Rx latches write data into the Bridge Control Register based on the state of lower byte enable only. The 80960Rx does not check the state of upper byte enable. This erratum applies to the Configuration Writes from the primary PCI interface as well as Local Bus writes from the i960 core processor.

IMPLICATION: A byte write to the lower byte of the Bridge Control Register causes data to be written to all 16 bits of the register, including the upper byte of the register (namely bits 8, 9 and 11). The data written to these bits is random. A byte write to the upper byte of the Bridge Control Register does not change the contents of the register.

WORKAROUND: Write the Bridge Control Register as a short word (16-bits).

6. Parity checking for inbound PCI address cycles is always enabled for the ATU

PROBLEM: The Parity Checking Enable bit (bit 06) in the primary and secondary ATU Command Registers (local bus address 1204H and 1298H) only affects inbound parity checking on PCI data cycles. Parity checking is always enabled for address cycles regardless of this bit's setting.

IMPLICATION: PCI masters that access 80960Rx local memory through the ATUs must generate address parity.

WORKAROUND: Make certain to connect the P_PAR and S_PAR signals from the 80960Rx to their respective PCI buses. Use PCI masters that generate address parity in all cases.

7. Configuration Write Cycle occurring simultaneously with clearing the Configuration Cycle Retry bit causes a deadlock problem

PROBLEM: In initialization modes 1 and 3, all PCI configuration cycles targeted at the 80960Rx on the primary PCI bus are retried until the Configuration Cycle Retry bit in the Extended Bridge Control Register (local bus address 1040H) is cleared by 80960Rx software. This allows software to setup the 80960Rx registers before the system host BIOS can configure it. The 80960Rx may enter a lockup condition which can only be cleared by a RESET when:

- the first configuration cycle to the 80960Rx is a write cycle which is being retried, and
- the Configuration Cycle Retry bit is cleared

The configuration cycle must be a write; configuration reads function properly when the Configuration Cycle Retry bit is cleared.

IMPLICATION: When used in applications where the first configuration cycle targeted at the 80960Rx is a write cycle, the 80960Rx may enter a lockup that can only be cleared by a device RESET. For PCI add-in card or motherboard applications, the system BIOS generates the initial configuration cycles.

WORKAROUND: For applications where the first configuration cycle targeted at the 80960Rx could be a write cycle, use initialization modes 0 or 2.

8. PCI Master Parity Error bits set in Bridge Interrupt Status Registers regardless of whether parity checking is enabled

PROBLEM: The PCI Master Parity Error bits (Bit 0) in both the Primary and Secondary Interrupt Status Registers (local bus addresses 1044H and 1048H) are set when a parity error occurs regardless of whether or not parity checking has actually been enabled. Parity checking is enabled and disabled for the primary side by bit 6 of the Primary Command Register (local bus address 1004H) and for the secondary side by bit 0 of the Bridge Control Register (local bus address 1040H).

IMPLICATION: An NMI# to the i960 core processor is generated when either of the PCI master Parity Error bits are set.

WORKAROUND: The user’s NMI# interrupt service routine should ignore NMI#s generated by PCI Master Parity Errors when parity checking has been disabled.

9. Deadlock condition under simultaneous inbound read and outbound write PCI traffic through secondary ATU

PROBLEM: A single 64-byte data queue in the secondary ATU is shared between inbound reads and outbound writes. This failure mode occurs when three conditions exist around the same internal clock edge:

1. A PCI master is reading data from the last location of the 64-byte data queue.
2. That same location is being filled from the 80960 local bus.
3. An outbound write occurs from the i960 core processor.

The internal queue control logic can potentially deadlock on these three conditions. The inbound read completes correctly but the outbound write never occurs. The 64-byte inbound read/outbound write queue is deadlocked and will accept no further transfers from either the secondary PCI bus or the 80960 local bus.

This failure mode only occurs on the secondary ATU; the primary ATU is not affected.

IMPLICATION: During heavy inbound read and outbound write traffic, the shared 64-byte inbound read/outbound write queue can deadlock. This condition can only be cleared by a device reset.

WORKAROUND: Three workarounds are:

1. Limit inbound PCI reads through the secondary ATU to 60 bytes or less.
2. Use the DMA controller to transfer data between the secondary PCI bus and the 80960Rx's local memory.
3. Don't perform simultaneous inbound reads and outbound writes through the secondary ATU.

10. Intermittent ONCE Mode upon power-up

PROBLEM: Intermittently, the device may enter ONCE Mode when excessive noise is present on the system's P_RST# signals. When in ONCE Mode the internal PLLs are stopped and all output signals are floated.

IMPLICATION: In some systems the power ramp and noise on the P_RST# signal may cause the device to enter ONCE Mode. This condition can only be cleared by device reset (cycling power off and on).

WORKAROUND: Ensure that the P_RST# signal is "noise-free" during the power ramp. ONCE Mode may be entered on through the JTAG port using the HIZ instruction. The ability to enter the ONCE mode by asserting the LOCK#/ONCE# pin during reset is disabled in the 80960RP 33/5.0 A-1 stepping.

11. ATU lockup condition under simultaneous outbound read data and inbound write data through the inbound data queue

PROBLEM: In each ATU, a 64-byte inbound data queue is shared by both outbound reads and inbound writes. A boundary condition causes an ATU deadlock when an outbound read from the i960 core processor begins during a bus master-initiated (IRDY# asserted) inbound write transaction. For this condition to occur, the inbound write's first address-to-data cycle must have more than three master inserted (IRDY#) wait states. Subsequent address-to-data cycles are not affected during the inbound write transaction. This problem can occur in either the primary or secondary ATU.

IMPLICATION: Under heavy use, the ATU deadlocks when using PCI masters that initiate inbound writes with greater than three master inserted (IRDY#) wait states. This condition can only be cleared by a device reset.

WORKAROUND: Three possible workarounds are:

1. Use bus masters that do not initiate writes through the inbound data queue that use greater than three master inserted (IRDY#) wait states.
2. Use the DMA channel to transfer data from the PCI bus to the 80960 local bus.
3. Do not perform simultaneous outbound reads and inbound writes through the ATUs.

12. PCI-to-PCI bridge can corrupt data during Memory Write and Invalidate cycles that insert IRDY# wait states during data-data transfers

PROBLEM: The 80960RP 33/5.0's integrated PCI-to-PCI bridge has 16 dword upstream and downstream posted write queues. The bridge corrupts data in the posted write queues when a PCI master writing into the queues inserts one or more IRDY# wait states in between the 15th and 16th dword during MWI cycles. This problem only affects MWI cycles; memory write cycles function correctly.

IMPLICATION: The PCI-to-PCI bridge corrupts data when used with PCI masters that insert IRDY# wait states during MWI cycles.

WORKAROUND: Two workarounds are:

1. Program the PCI-to-PCI bridge cacheline size register (Local bus offset 0x100CH) to 0. This aliases all MWI commands and memory write commands. This workaround can limit write performance when used with host bridges that are optimized for MWI commands.
2. Do not use PCI masters that insert IRDY# wait states during MWI commands.

13. Using the Messaging Unit circular queues with more than one recovery wait state on the local bus can cause primary Address Translation Unit data corruption

PROBLEM: When using the Messaging Unit (MU) circular queues with more than one recovery wait state on the 80960 local bus, data corruption in the primary ATU occurs. Specifically, inbound reads through the primary ATU return incorrect data, and inbound writes never propagate to 80960RP 33/5.0 local memory. This errata only affects the circular queue functionality of the MU: no other messaging unit functionality (i.e., doorbell registers, index registers, etc.) is affected.

IMPLICATION: Data corruption can occur when using the circular queues and the primary ATU when more than one recovery wait state is inserted during 80960 local bus cycles.

WORKAROUND: Never insert more than one additional recovery wait state during any 80960 local bus cycle when using the circular queues and the primary ATU. Note that the 80960RP 33/5.0 explicitly inserts one recovery cycle into every 80960 local bus access. This workaround is for additional recovery states. For designs that use the on-chip memory controller, this can be done by programming the proper device registers. The following bit locations in these registers affect recovery cycles and must be programmed to a 0 or 1:

- Memory Bank Read Wait States Register bits 2:0 (MBRWS1:0 at offset 1508H, 1514H)
- Memory Bank Write Wait States Register bits 2:0 (MBWWS1:0 at offset 150CH, 1518H)
- DRAM Bank Read Wait State Register bits 1:0 (DRWS at offset 1524H)
- DRAM Bank Write Wait State Register bits 1:0 (DWWS at offset 1528H)

When programming the Memory Controller for interleaved FPM DRAM, the number of write cycle CAS# delays (T_{WCP}) adds to the number of recovery states. For example, in a typical interleaved FPM design, T_{WCP} is programmed for 1.5 cycles (bits 9:8 in the DWWS register set to 00 or 01). One additional write recovery wait state is also inserted. In this case, the write recovery wait states must be cleared to 0 (bits 1:0 in the DWWS register cleared to 00) for this errata. When programmed for zero additional recovery cycles, the memory controller de-asserts RAS# for two clock cycles to satisfy the RAS# pre-charge specification of the DRAM.

14. Access to non-existent DRAM causes incorrect error reporting

PROBLEM: An 80960 local bus fault error occurs when the i960 core processor attempts to access an invalid local bus memory address and the Bus Monitor Interrupt (BMER register at local bus address 1534H) is enabled. The next valid inbound ATU access occurring after the local bus fault causes a target abort on the PCI bus. The errors reported in the Primary ATU Interrupt Status Register are: 80960 Bus Fault and PCI Target Abort (target).

IMPLICATION: A valid inbound ATU transaction incorrectly generates a PCI target abort when a bus monitor interrupt occurs.

WORKAROUND: Do not enable the Bus Monitor Interrupt feature. To disable this feature, clear bit 0 of the Bus Monitor Enable register, which is located at local bus offset 1534H.

15. Address Translation Unit write and Messaging Unit queue port write may get out of order on the 80960 local bus

PROBLEM: An inbound ATU data write and an inbound MU queue port write can get out of order on the 80960 local bus. The MU queue port write may complete before the ATU data write completes on the local bus, regardless of whether the MU queue port write began after the ATU data write on the PCI bus.

An interrupt is generated to the i960 core processor when the write to the MU queue port occurs. Because the ATU write could still be pending in the 80960Rx inbound write buffer, the interrupt service routine could potentially read bad data from local memory if it depended on the ATU write completing before the MU queue port write.

IMPLICATION: Invalid data could be read from the 80960Rx local memory if a user application depended on the 80960Rx maintaining order between ATU and messaging unit writes.

WORKAROUND: To prevent the i960 core processor from reading the message data, before the ATU completes the write, program the Local Bus Arbitration Latency Counter Register (LBALCR) to a value large enough to prevent the core processor from accessing the Local Bus between the two transactions. The value needed in the LBALCR must handle both the Message Unit transaction, the ATU transaction, and a potential DRAM refresh cycle. Use the following equation to calculate the minimum LBALCR value:

LBALCR = 26 + (2 * TRC) + (16 * (1 + TCP)) + (2 * TRCV)

where:

TRC is the number of wait states between address and the first data.

TRC = TRRC + 2 (T_{RRC} is the DRAM read cycle RAS-to-CAS delay), or

$T_{RC} = T_{WRC} + 2$ (T_{WRC} is the DRAM write cycle RAS-to-CAS delay)

T_{CP} is the number of wait states between data during a burst.

$T_{CP} = T_{RCP}$ (T_{RCP} is the DRAM read cycle CAS pulse width), or

$T_{CP} = T_{WCP}$ (T_{WCP} is the DRAM write cycle CAS pulse width)

T_{RCV} is the number of recovery cycles at the end of a DRAM access.

$T_{RCV} = T_{RRCV} + 1$ (T_{RRCV} is the DRAM read cycle additional recovery wait states), or

$T_{RCV} = T_{WRCV} + 1$ (T_{WRCV} is the DRAM write cycle additional recovery wait states)

These values are found in the DRAM Bank Read Wait State (DRWS) and DRAM Bank Write Wait State (DWWS) registers. The 26 cycles includes 20 cycles for a worst case DRAM refresh cycle.

For example:

$T_{RRC} = T_{WRC} = 1$; $T_{RC} = 3$ (3 wait states from address to data)

$T_{RCP} = T_{WCP} = 0$; $T_{CP} = 0$ (0 wait states between data)

$T_{RRCV} = T_{WRCV} = 1$; $T_{RCV} = 1$ (1 extra recovery cycle after DRAM access)

Then,

LBALCR = 26 + (2 * 3) + (16 * (1 + 0)) + (2 * 1) = 50 = 32H

16. Using multiple DMA channels that perform unaligned transfers can cause a DMA channel performing a read to stop operating

PROBLEM: When using two or more DMA channels simultaneously — one DMA channel is performing a read operation, and at least one of the other DMA channels is unaligned — the DMA channel performing a read may terminate its task. Aligned means the source address and destination address are both on the same byte boundary.

IMPLICATION: The DMA channel performing the read terminates and is no longer operational.

WORKAROUND: You may use any of the following workarounds:

1. Use only one DMA channel at a time.
2. Operate all DMA channels aligned.
3. If the DMA channels are only performing writes, the channels can be aligned or unaligned.
4. If multiple DMA channels are used, one read channel can be aligned or unaligned if the other channel(s) is(are) performing aligned writes.

17. Messaging Unit logs multiple interrupt sources when using the index registers

PROBLEM: Accesses to an index register within the Messaging Unit (MU) cause multiple interrupt status bits to be set in the Inbound Interrupt Status Register (IISR). The bits are the Index Register Interrupt (bit 06), Inbound Doorbell Interrupt (bit 02), Inbound Message 1 Interrupt (bit 01), and Inbound Message 0 Interrupt (bit 00). This occurs when the Inbound Interrupt Mask Register's Index Register Interrupt Mask bit is set or cleared.

IMPLICATION: 80960RP software cannot correctly determine the source of the interrupt in the MU when the Index Registers are used with another function in the MU.

WORKAROUND: Do not use the index registers with either:

- Doorbell registers and Message Queues
- Message registers and Message Queues

18. DMA Descriptors appended to the end of a chain may not execute

PROBLEM: A descriptor appended to a DMA chain may not execute when the Chain Resume bit (bit 01) is set in the Channel Control Register. This occurs when:

1. The last descriptor of the existing chain is a DMA read, and
2. The Chain Resume bit is set when the last word of the DMA is being transferred.

When condition 1 and 2 occur, the DMA unit does not re-read the Next Descriptor Address (NDA) of the current descriptor.

This erratum exists for both aligned and unaligned DMA transfers.

IMPLICATION: A DMA transfer from an appended DMA descriptor may not execute.

WORKAROUND: Two workarounds can be used to prevent this errata:

1. Add a NULL descriptor to the end of a chain where the last descriptor is a read. This applies to original chains and to appended chains even when the appended chain is one descriptor in length. The NULL descriptor has a Byte Count = 0000H, and an NDA of 0000H. A NULL descriptor at the end of a DMA chain is appended in the normal manner — the NDA of the last descriptor of the existing chain is changed to point to the new chain — then the Chain Resume bit is set.
2. Append chains as normal, then poll the state of the Channel Active Flag (bit 10) in the Channel Status Register. When the flag is cleared, set the Chain Resume bit once more.

19. Downstream exclusive read transaction through multiple bridges may cause a deadlock condition

PROBLEM: A deadlock condition is possible with an 80960Rx behind another 80960Rx during a downstream exclusive read transaction. The condition occurs when the upstream bridge attempts the exclusive access on its secondary PCI bus and the downstream bridge has a posted memory write (PMW) transaction in its queue. The ordering rules require the downstream bridge to complete the upstream PMW before accepting the downstream read transaction. The upstream bridge, however, retries all upstream transactions once it begins the exclusive downstream transaction, and retries the downstream bridge's PMW, thus causing a deadlock condition.

IMPLICATION: Under the conditions outlined above, the 80960Rx bridge unit may deadlock.

WORKAROUND: To prevent this erratum do not perform locked PCI transactions to PCI devices downstream of the 80960Rx.

20. Inbound ATU PCI read issue with retried completion cycles

PROBLEM: For inbound ATU read requests, both primary and secondary ATUs implement delayed transactions per the *PCI Local Bus Specification*, revision 2.1. When a PCI initiator attempts to read from an inbound ATU, the initial request is retried while the ATU begins reading the data from the 80960 local bus. Once the data is in the inbound read queue, the initiator completes the transaction during the delayed completion cycle. The ATU can deadlock when a second initiator attempts to read the same address location with a different transfer length before the first master returns from its retry cycle.

IMPLICATION: When two PCI initiators are accessing the same address location with different length data requests, the ATU can potentially deadlock. The ATU only recovers from this condition with a device RESET (P_RST#).

WORKAROUND: Ensure that your 80960Rx application does not allow two PCI masters accessing the same memory address location simultaneously through either the primary or secondary ATU.

21. Null writes (BE3:0# deasserted) to the MU Inbound and Outbound Queue Ports corrupt the corresponding queue pointer

PROBLEM: A *null write* is defined as a write cycle on the PCI bus when all Byte Enables are deasserted; no data is actually transferred. When a null write occurs to either the Inbound or Outbound Queue port, the queue pointers are incremented. These queue ports are part of the Messaging Unit and are located at offset 0040H and 0044H in the first 4 Kbytes of the Primary Inbound ATU PCI Address Space.

IMPLICATION: The queue pointers are corrupted when a host bridge or another PCI device performs a null write to either the Inbound or the Outbound Queue ports.

WORKAROUND: Always perform only 32-bit DWORD writes to the both the Inbound Queue Port and the Outbound Queue Port in the Messaging Unit.

22. PCI-to-PCI Bridge Unit can corrupt data during Memory Write cycles when wait states are inserted simultaneously on both the primary and secondary sides of the bridge

PROBLEM: The 80960RP’s integrated PCI-to-PCI bridge has 16-DWORD upstream and downstream posted write queues. The bridge can corrupt data in the posted write queues when a PCI initiator writing into the queues inserts IRDY# wait states between the 15th and 16th DWORD at the same time that the target inserts TRDY# wait states on the other side of the bridge during Memory Write cycles.

IMPLICATION: The PCI-to-PCI bridge corrupts data when used with PCI initiators and corresponding targets that insert wait states during memory write cycles.

WORKAROUND: Either of two workarounds can prevent this erratum:

- Do not use PCI masters that insert IRDY# wait states during Memory Write commands.
- Disable write posting: clear bit 00 of the Extended Bridge Control Register (EBCR), local bus address 1040H.

SPECIFICATION CHANGES

1. PCI Interrupt Routing Select Register (PIRSR) Polarity

ITEM: The polarity of the XINT Select bit (bit 0) in the PCI Interrupt Routing Select Register (PIRSR) has changed. Previous to the 80960RP 33/5.0 A-0 stepping, it was specified as:

- 0 - Interrupts routed to i960 core processor interrupt controller input. This is the default condition when the 80960RP 33/5.0 is reset.
- 1 - Interrupts routed to P_INTX# pins.

In the 80960RP 33/5.0 A-0 stepping, the specification is changed to:

- 0 - Interrupts routed to P_INTX# pins. This is the default condition when the 80960RP 33/5.0 is reset.
- 1 - Interrupts routed to i960 core processor interrupt controller input.

This allows the 80960RP 33/5.0 to function as a stand-alone bridge while the core is held in reset as in Initialization Mode 0.

2. Asynchronous clocking mode not supported

ITEM: The *i960® RP Microprocessor User's Manual* (272736-001) describes an asynchronous clocking mode which allows the secondary PCI bus and the i960 core processor to run at an asynchronous and slower frequency than the primary PCI bus. This mode of operation is not supported. Synchronous mode is the only clocking mode specified.

For proper operation, the WIDTH/HLTD0/SYNC pin should either remain unconnected or pulled high with a 10 K ohm pullup resistor. The P_CLK pin should be tied to V_{CC} or V_{SS}.

3. The MU interlock mechanism remains enabled for APIC registers when the APIC is disabled

ITEM: On the first access by a PCI master to the APIC registers in the Messaging Unit, the interlock mechanism is expected to generate an interrupt to the i960 core processor. If the PCI master returns and writes to an APIC register, the MU will retry the master until the interrupt is cleared. If the APIC is disabled, the interrupt is never generated, and thus never cleared, causing a PCI master lockup condition.

Do not access the two APIC registers in the MU unless the APIC Unit is enabled. These registers are: APIC Register Select Register (ARSR) and APIC Window Register (AWR).

4. *Incorrect BIOS access of block size from Base Address Registers*

ITEM: To determine the block size requirements of a PCI device, the system BIOS is to write all ones (FFFF FFFFH) to the base address register then read back the register. The returned value indicates the block size requirements based on which bits are set to one (i.e., the memory required for the PCI device). When performing this operation, some system BIOS' write FFFF FFFEH to the base address register.

To determine the proper block size requirement, a value of FFFF FFFFH is written to the base address register. The next read after the FFFF FFFFH write is then directed to the limit register instead of the base register; the limit register is then used to determine the block size required.

The specification is changed to define that either of the following values can be used to determine the block size for the device:

- all ones (FFFF FFFFH)
- all ones except bit 0 (FFFF FFFEH)

This affects the Primary Inbound ATU Base Address Register, the Expansion ROM Base Address Register, and the Secondary Inbound ATU Base Address Register.

See also Documentation Change for Section 16.7.14.1, Page 16-38.

5. *ATU Configuration Register Bit 12 Definition is Changed*

ITEM: The ATU Configuration Register (ATUCR) Bit 12 was previously defined as “Reserved”; it is now defined as “Secondary Bus, Messaging Unit Access Enable”. See definition below; see also the Documentation Change for Section 16.7.32, Page 16-51, Table 16-43.

Bit	Default	R/W	Description
12	0_2	Read/Write	Secondary Bus, Messaging Unit Access Enable - When set, the PCI-to-PCI Bridge unit can forward a transaction from the secondary PCI interface, through the bridge, and to the Messaging Unit (first 4 Kbytes of the PATU inbound address space) on the primary PCI interface. For correct operation, the transaction must be a valid bridge address (claimed by secondary interface of the bridge and forwarded to the primary interface of the bridge) as well as a valid Messaging Unit address. When clear, the Messaging Unit cannot claim a transaction mastered by the primary interface of the bridge.

6. Multiple reads of the Base Address Register after writing all 1's will return different values (See Specification Clarification Item #5)

(In previous revisions of this document, this item was redundant with a Specification Clarification.)

7. Additional devices can be configured as public or private on the secondary PCI bus by programming the Secondary IDSEL Select Register (SISR)

ITEM: Bits 5 through 9 of the Secondary IDSEL Select Register (SISR) now can be used to mask the SAD21 through SAD25 during type 1 to type 0 configuration cycles on the secondary bus. This enables up to 10 devices to be configured as public or private devices on the secondary PCI bus.

8. PCI Interrupt Routing Select Register (PIRSR) now supports individual routing of each XINT3:0# pin

ITEM: Bits 3:0 of the PCI Interrupt Routing Select Register (PIRSR) individually control the routing of the XINT3:0# pins to either the i960 core processor or the corresponding P_INTx# pin:

- Setting a bit to a 1 routes the corresponding interrupt to the i960 core processor.
- Clearing the bit to a 0 routes the corresponding interrupt to the P_INTx# pin.

9. The Memory Bank Extended MWE3:0# bits in the Memory Bank Control Register can provide one clock of address hold time during write cycles

ITEM: The description for both Memory Bank 1 Extended MWE3:0# bit and Memory Bank 0 Extended MWE3:0# bit should now read:

"This bit field enables or disables extending the deassertion period for the MWE3:0# signal during burst write cycles. The bit also enables one clock of MA11:0 and BE1:0 hold time relative to the rising edge of MWE# during writes to this region.

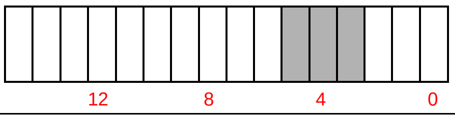
- When cleared (0), deassertion period is one-half of a CLKIN period.
- When set (1), the deassertion period is extended by the wait state profile defined in the MBWWSx registers in addition to the one-half clock in period. Also when set, the MA11:0 and BE1:0 keep their current state for one clock after MWE3:0# are deasserted. This also adds an extra wait state."

10. Mask bits added for all PCI-PCI bridge error conditions which may cause an NMI# to the i960 core processor

ITEM: Bits 15 through 4 of the Secondary Decode Enable Register (SDER) now can be used to mask sources of NMI# from the bridge. When set to 1, the source of NMI# is masked. When cleared to 0, the source of NMI is enabled. See the following table:

Table 10-1. Secondary Decode Enable Register - SDER

PCI Configuration Address Offset: 5CH
80960 Core Local Bus Address: 105CH



Bit	Default	R/W	Description
15	**0_2**	**R/W**	**Secondary SERR# Detected (Bit 4 of SBISR)**
14	**0_2**	**R/W**	**Secondary PCI Master Abort (Bit 3 of SBISR)**
13	**0_2**	**R/W**	**Secondary PCI Target Abort (Initiator, Bit 2 of SBISR)**
12	**0_2**	**R/W**	**Secondary PCI Target Abort (Target, Bit 1 of SBISR)**
11	**0_2**	**R/W**	**Secondary PCI Master Parity Error (Bit 0 of SBISR)**
10	**0_2**	**R/W**	**Primary SERR# Asserted or Detected (Bit 4 of PBISR)**
09	**0_2**	**R/W**	**Primary PCI Master Abort (Bit 3 of PBISR)**
08	**0_2**	**R/W**	**Primary PCI Target Abort (Initiator, Bit 2 of PBISR)**
07	**0_2**	**R/W**	**Primary PCI Target Abort (Target, Bit 1 of PBISR)**
06	**0_2**	**R/W**	**Primary PCI Master Parity Error (Bit 8 of PSR)**
05:03	**000_2**	**Read Only**	**Reserved**
02	0_2	Read/ Write	Private Memory Space Enable - when set, this bit disables Bridge forwarding of addresses in the SMBR/SMLR address range. This creates a private memory space on the secondary PCI bus for peer to peer transactions.
01	0_2	Read/ Write	Secondary Positive Memory Decode Enable - when set, this bit enables the secondary interface of the bridge unit to positively decode memory addresses on the secondary bus. Addresses within the SMBR/SMLR address range is forwarded through the bridge. Inverse decoding is disabled.
00	0_2	Read/ Write	Secondary Positive I/O Decode Enable - when set, this bit enables the secondary interface of the bridge unit to positively decode I/O addresses on the secondary bus. Addresses within the SIOBR/SIOLR address pair is forwarded through the bridge. Inverse decoding is disabled.

11. Multiple reads of the Base Address Register, after writing all 1's, returns the limit register value until rewritten

ITEM: The 80960Rx provides a programmable mechanism for defining the memory address block size requirements. This mechanism uses the Base Address Register (BAR) and corresponding limit register. 80960Rx initialization code programs into the limit register the desired value to be returned for memory block size. To determine the memory block size requirements, write FFFF FFFFH or FFFF FFFEH to the BAR, then read back the BAR value. Subsequent reads of the BAR returns the memory block size (i.e., it returns the limit register value until the BAR is rewritten with a value other than FFFF FFFH or FFFF FFFEH).

SPECIFICATION CLARIFICATIONS

1. Five IDSEL lines for the secondary PCI bus are not enabled

ITEM: Bits 5 through 9 in the Secondary IDSEL Select Register (SISR); see page 15-58 of the *i960® RP Microprocessor User's Manual* (272736-001), may be used in future versions to enable the masking of S_AD21:25 during Type 1 to Type 0 configuration cycles on the secondary PCI bus. These five signals (in addition to S_AD16:20) will enable additional devices as public or private PCI devices in an intelligent I/O system.

2. How to use the Data Enable (DEN#) Signal with an In-Circuit Emulator

ITEM: When using an ICE in your 80960Rx system, the use of the Data Enable signal (DEN#) is not recommended. This clarification describes how DEN# operates and a recommended solution for using it with an In-Circuit Emulator (ICE).

DEN# Operation: When asserted, DEN# indicates data transfer cycles during a bus access. DEN# asserts at the start of the first data cycle in a bus access and de-asserts at the end of the last data cycle. DEN# can be used in conjunction with DT/R# to provide control for data transceivers connected to the data bus.

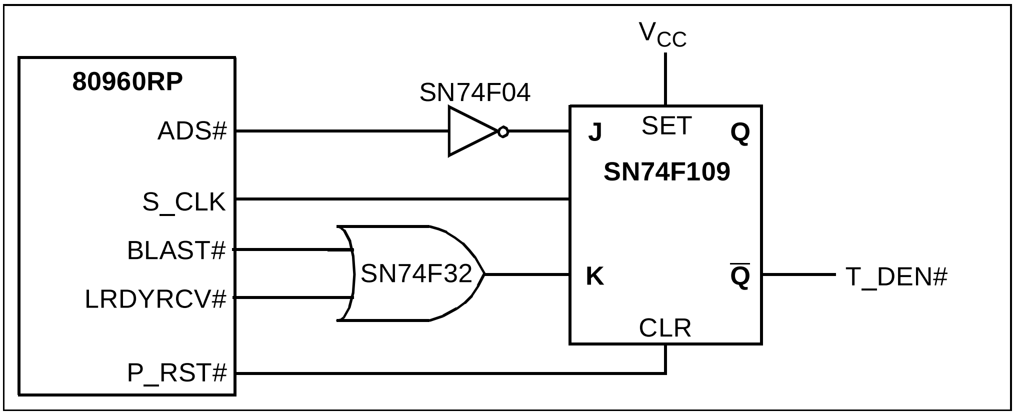
Using DEN# with an In-Circuit Emulator: For ICE users, it is not recommended that you use the 80960Rx's DEN# signal directly to your transceivers**.** When executing an ICE microcode transaction, the expected behavior is that DEN# would remain de-asserted during the entire transaction. However, DEN# asserts as described above. If your design uses DEN# to enable transceivers, the transceivers will be enabled. This may result in bus contention.

The 80960Rx is highly integrated; for most designs, the use of transceivers on the 80960 local bus is not necessary.

The use of DEN# in 80960Jx designs was possible because the 80960Jx was on a "POD". The POD was cabled to your target board where it plugged in to a socket. The POD could mask out DEN# during ICE microcode transactions. The 80960Rx's package does not allow the use of a POD; consequently, the ICE signals connect directly to the target system and the DEN# signal cannot be masked.

WORKAROUND: To use an ICE with your 80960Rx design, alternatives to DEN# are:

- Ground the OE# pin of the transceiver
- Re-create a DEN# signal with the circuit shown below



The circuit asserts T_DEN# (Q#) at the start of the first data cycle when ADS# asserts and BLAST# and LRDYRCV# de-asserts. T_DEN# deasserts at the end of the last data cycle when ADS# de-asserts and BLAST# and LRDYRCV# assert. During RESET, T_DEN# de-asserts.

Equivalent components may be used in place of the components shown.

3. Accessing 80960Rx MMRs via the primary PCI bus

ITEM: The system cannot access Memory-Mapped Registers (MMRs) at 80960 local bus addresses 1000H through 104CH via primary inbound ATU translation. (The default secondary Inbound ATU translation to the 80960Rx MMRs is not affected; it operates as specified.)

The Primary Inbound ATU Translate Value Register (PIATVR) default value is 1000H. When this maps into 80960 local bus address space, the MMRs at locations 1000H through 104CH are not accessible by a primary PCI bus master. MMRs in the range from 1050H through 1FFCH are accessible through the Messaging Unit index registers.

MMRs located from 1000H through 104CH are part of the 80960Rx's Bridge Configuration Header and are accessible via Type 0 PCI Configuration Cycles from the primary PCI interface.

4. Parity on data cycles not checked before delayed write completion cycles are accepted

ITEM: I/O writes and configuration cycle writes are treated as delayed transactions through the 80960Rx's PCI-to-PCI bridge. A delayed write cycle is first retried on the initiating bus while the write propagates through the bridge to the target bus. When the write cycle completes on the target bus, the retried cycle on the initiating bus is accepted; this is called the delayed write completion cycle.

Parity is checked when the cycle is first accepted on the initiating bus and errors reported correctly. If a parity error occurs at this time, the 80960Rx asserts PERR# and does not propagate the write through to the target bus. During the delayed write completion cycle, the 80960Rx accepts the cycle and ends the delayed write, regardless of whether or not data parity was correct.

When a parity error occurs during the delayed write completion cycle, the 80960Rx still asserts the PERR# pin correctly. Note that because this is the delayed write completion cycle, correct parity is still delivered to the device on the target bus.

5. Multiple reads of the Base Address Register after writing all 1's will return different values

ITEM: The 80960Rx provides a programmable mechanism for defining the memory block size requirements. This mechanism uses the Base Address Register (BAR) and corresponding limit register. 80960Rx initialization code programs into the limit register the desired value to be returned for memory block size. To determine the memory block size requirements, write FFFF FFFFH or FFFF FFFEH to the BAR, then read the BAR. On the first read, this value is the memory block size (i.e., the limit register value); all subsequent reads of the BAR will return a value other than the memory block size.

6. When determining memory address block size, accesses to the Base Address Register must be 32-bit configuration cycles

ITEM: When determining block size requirements, the 80960Rx's Base Address Register (BAR) must be accessed by 32-bit configuration cycles. Writing FFFF FFFFH or FFFF FFFEH to the BAR must be performed as a 32-bit configuration write cycle. Reading the BAR, to determine the block size requirements, must be a 32-bit configuration read cycle.

Configuration cycles not used to determine block size requirement can be performed as 8-, 16-, or 32-bit cycles.

7. Secondary PCI bus reset notification to i960 core processor

ITEM: The 80960Rx's PCI-to-PCI Bridge Unit provides for generation of a software reset to the secondary PCI bus. Setting the Bridge Control Register's Secondary Bus Reset bit (bit 06 = 1) causes the S_RST# output (ball J25) to assert. This resets devices on the secondary PCI bus as well as specified 80960Rx registers, queues, and units. Clearing the bit results in the S_RST# output to deassert.

The 80960Rx has no internal mechanism to automatically detect when a secondary bus reset has occurred.

WORKAROUND: When application software requires either the re-initialization of PCI devices on the secondary PCI bus or re-programming of the affected 80960Rx registers, two possible workarounds are:

1. Connect the 80960Rx's S_RST# output to the 80960Rx's XINT4# input (ball P2). Set bit 04 in the Interrupt Mask Register (local bus address FF00 8504H) and program bits 03:00 in the Interrupt Map Register 1 (local bus address FF00 8524H) to the desired vector. The interrupt service routing for XINT4# checks the state of the Secondary Bus Reset bit. When this bit is cleared, the reset is complete. The Interrupt Service Routine (ISR) can then initialize the secondary PCI devices and 80960Rx registers.
2. After the secondary bus is reset, the host BIOS runs PCI configuration cycles to reinitialize any public PCI devices. Configuration cycles to non-existent devices generate master aborts on the secondary bus, which causes an NMI# to the 80960Rx. A software workaround is to add a check of the Secondary Bus Master Abort interrupt service routine to confirm when the secondary ATU registers are reset.

   The NMI interrupt service routine — when finding the NMI Interrupt Status Register's Secondary Bridge Error bit (bit 04) is set (local bus address 1700H) — reads the Secondary Bridge Interrupt Status Register (SBISR) (local bus address 1048H). In the SBISR, the PCI Master Abort bit (bit 03) is set in the event of a Master Abort occurring on the 80960Rx's secondary PCI interface. When this bit is set, the ISR checks the contents of the secondary ATU registers that were originally programmed by the initialization code.

   — When the register(s) is reset, the ISR re-initializes the secondary ATU registers and any private PCI devices on the secondary PCI bus.

   — When the register is not reset, the ISR can perform the existing servicing of the master abort.

   One bit that may be used as a check is the Secondary ATU Command Register's Memory Enable bit (bit 01 - local bus address 1298H).

See also Documentation Change: Item 39., Section 15.13.23, Page 15-54, Table 15-35 (pg. 43).

8. *Parity error reporting during configuration cycles*

ITEM: The enabling and reporting of parity handling during Type 0 bridge configuration cycles is contained within the primary ATU. The primary ATU acts as a proxy for the bridge by claiming Type 0 configuration cycles to the bridge (Function 0). The primary ATU then records the errors to its own 80960 local bus MMRs.

After device reset, parity handling is disabled in the bridge and primary ATU command registers. The following scenarios can occur during configuration cycles:

1. Bridge Parity Handling Enabled - Parity handling and SERR# assertion is enabled in the bridge but not in the primary ATU. When an address parity error is detected during a subsequent bridge configuration cycle, the parity errors *are not* recorded (i.e., SERR# is not asserted for address parity and PERR# is not asserted for configuration writes).
2. Primary ATU Parity Handling Enabled - Parity handling and SERR# assertion is enabled in the primary ATU and not in the bridge. When an address parity error is detected during a subsequent bridge configuration cycle, the parity errors *are* recorded in the Primary ATU Status Register (PATUSR) (i.e., SERR# is asserted for address parity and PERR# is asserted for configuration writes).
3. Bridge and Primary ATU Parity Handling Enabled - Parity handling and SERR# assertion in both the bridge and the primary ATU is enabled. Address parity errors are reported with SERR# and PERR#, but the errors are only recorded in the PATUSR — not in the bridge status register (Primary Status Register - PSR).

DOCUMENTATION CHANGES

1. Section 8.2.2, Page 8-16

First paragraph, first sentence incorrectly states: "The XINT7:0# pins and the NMI# pin use level-low detection." It now correctly states: "The XINT7:0# pins use level-low detection."

2. Section 8.3.3, Page 8-27

The last sentence of the first paragraph is changed to read: "The NMI interrupt latch is cleared by clearing the sources of all interrupts at the internal peripherals. A new edge triggered interrupt is generated to the i960 core processor only after all interrupt status bits are simultaneously clear."

3. Section 8.4.7, Page 8-36

The third sentence of the third paragraph is changed to: "In addition, software must check the contents of the NISR to ensure all NMI sources are cleared before returning from the NMI interrupt service routine." Also, the following code sample is included in this section.

Example Code - NMI Interrupt Handler Main Loop

```
/*  NMI Interrupt Handler */
volatile unsigned long int NISR;
do
     {   NISR = *NISR_reg_addr;
if (NISR & 1)
     80960_core_error();
if (NISR & 2)
     primary_atu_error();
if (NISR & 4)
     secondary_atu_error();
if (NISR & 8)
     primary_bridge_interface_error();
if (NISR & 16)
     secondary_bridge_interface_error();
if (NISR & 32)
     dma_channel_0_error();
if (NISR & 64)
     dma_channel_1_error();
if (NISR & 128)
     dma_channel_1_error();
if (NISR & 256)
     messaging_unit_interrupt();
if (NISR & 512)
     extnernal_nmi_interrupt();  }
while( !NISR );
return;
```

4. Section 8.4.7, Page 8-37, Table 8-10

The correct definition for bits 8 and 9 follows:

Bit	Default	Read/Write	Description
09	02	Read Only	External NMI# Interrupt - when set, an interrupt is pending on the external NMI# input. When clear, no interrupt exists.
08	02	Read Only	Messaging Unit Interrupt - when set, an NMI interrupt or error exists in the Messaging Unit. When clear, no error exists.

5. Section 9.10.8, Page 9-34

Added list of privileged instructions to first bullet. It now reads:

A TYPE.MISMATCH fault is generated when attempts are made to:

- Execute a privileged (supervisor-mode only) instruction while the processor is in user mode. Privileged instructions on the i960 RP processor are:

modpc	intctl
sysctl	inten
icctl	intdis
dcctl	

6. Section 11.2.1, Page 11-3

In the second paragraph on page 11-3, references to “asserted” should say “de-asserted”. The corrected paragraph now reads:

The RETRY signal is sampled on the rising edge of P_RST#. The value of this signal is written to the Configuration Cycle Disable bit in the EBCR. When RETRY is active and P_RST# is **de-asserted**, the 80960Rx signals a Retry on all PCI configuration cycles it receives on the primary PCI bus. When RETRY is inactive and P_RST# is **de-asserted**, the 80960Rx accepts PCI configuration cycles on the primary PCI bus.

7. Section 11.2.8, Page 11-6

The second paragraph incorrectly refers to Table 11-2; these references are removed. Also in the second paragraph: the reference to P_CLK should be S_CLK.

8. Section 11.4, Page 11-12

Last paragraph, last sentence: replace "...a system's..." with "...the 80960Rx's local bus...". The sentence now reads: "These stacks must be located in **the** 80960Rx**'s local bus** RAM."

9. Section 11.6, Page 11-24

First paragraph, third sentence: remove "P_CLK" from list of signals.

10. Section 11.6.1, Page 11-24

Replace the first paragraph with: "The i960 RP processor has a single clock input (S_CLK) for control. All input/output timings are relative to S_CLK."

Keep the second paragraph, delete the third and fourth paragraphs.

Delete Table 11-8.

Fifth paragraph: remove references to P_CLK. It now reads: "The clock input is designed to be driven by most common TTL crystal clock oscillators. The clock input must be free of noise and conform with the specifications listed in the *i960® RP I/O Processor at 5 Volts* Data Sheet. S_CLK input capacitance is minimal; for this reason, it may be necessary to terminate the S_CLK circuit board traces at the processor to reduce overshoot and undershoot.

11. Section 13.3.7, Page 13-18

Add this statement to follow the third paragraph: "The RDYRCV# signal is generated internally by the 80960Rx for accesses by the memory controller and does not have to be generated externally."

12. Section 13.6, Page 13-24

Add this statement to follow the first paragraph: "External bus masters do not have access to the 80960Rx's internal local bus. Therefore, an external bus master cannot access any of the 80960Rx's internal peripherals (e.g., the Memory Controller, the i960 core processor, etc.)."

13. Section 13.6.1, Page 13-26

Remove paragraph (following Figure 13-13). Replace with the following paragraph:

"The 80960Rx arbitration logic enables external bus masters to control 80960Rx local bus. The Local Bus Arbitration Unit maintains the basic 80960Rx protocol for the HOLD/HOLDA except that the 80960Rx processor will not respond to the assertion of the

HOLD signal (i.e., assert the HOLDA signal) during reset. This includes Processor Reset and Local Bus Reset.”

14. Section 14.4.1, Page 14-8, Table 14-2

In Table 14-2, the Bit 18 description is changed to:

Bit	Default	R/W	Description
18	0_2 Extension Disabled	Read/ Write	**Memory Bank 1 Extended** MWE3:0# **Bit** - This bit field enables or disables extending the deassertion period for the MWE3:0# signal during burst write cycles. When cleared (0), the deassertion period is one-half of a S_CLK period. When set (1), the deassertion period is extended by the wait state profile defined in the MBWWS1 register in addition to the one-half a S_CLK period.

In Table 14-2 on page 14-9, the Bit 2 description is changed to:

Bit	Default	R/W	Description
2	0_2 Extension Disabled	Read/ Write	**Memory Bank 0 Extended** MWE3:0# **Bit** - enables or disables the extension of the deassertion period for the MWE3:0# signal during burst write cycles. When cleared (0), deassertion period is one-half a CLKIN period. When set (1), the deassertion period is extended by the wait state profile defined in the MBWWS0 register in addition to the one-half a CLKIN period.

15. Section 14.4.5, Page 14-14, Figure 14-5

The MA11:0 signal shows five transitions. The last transition is incorrectly shown in the T_A cycle. It is now correctly shown in the T_R cycle (one cycle earlier).

16. Section 14.4.5, Page 14-15, Figure 14-6

The MA11:0 signal shows five transitions. The last transition is incorrectly shown in the T_A cycle. It is now correctly shown in the T_R cycle (one cycle earlier).

Also, above the T_R, t_{WWD}=0 is changed to t_{WWR}=0.

17. Section 14.4.5, Page 14-16, Figure 14-7

Data cycle labels modified to reflect other figure labels using upper case “T” with upper case subscript. Example: “T_A, T_D, T_R, T_W,”.

18. Section 14.4.5, Page 14-16, Figure 14-8

The signal labeled WRITE_CAS# is incorrectly named. It is changed to MWE3:0#.

19. Section 14.4.5, Page 14-17, Figure 14-9

The figure incorrectly shows MA11:0 transitioning in the second cycle (T_W cycle). This transition is removed; the next transition now occurs in the fifth cycle (the T_{WWD}=1 T_W cycle).

20. Section 14.5, Page 14-18, Table 14-6

Text added to the descriptions for MA11:0 now reads: “Memory Address Bus **- Specifies address path to the DRAM**.”

In the description of DP3:0, the first bullet should not be a bullet; it should be part of the DRAM Data Parity description.

The following sentence is added to the descriptions for LEAF1:0# and DWE1:0#: “For non-interleaved operation, these signals are identical and can be used interchangeably.”

21. Section 14.5.1, Page 14-22, Table 14-8

The max value for the 1 Mbit (256Kx4) and Non-interleaved DRAM is incorrectly shown as 1. The correct value is 4.

The min value for the 4 Mbit (256Kx16) and Non-interleaved DRAM is incorrectly shown as 4. The correct value is 1.

22. Section 14.5.4, Page 14-24, Table 14-9

Default for bits 7:3 incorrectly stated as “000 00”, corrected to state “0000 0”. Also changed bit alignment in description column.

In bit 2:1 description, in the “00” condition, remove **“(interleave FPM DRAM only)”**

23. *Section 14.5.6, Page 14-27, Table 14-11*

In the description for Bits 9:8, DRAM Read cycle CAS pulse width (t_{RCP}), the bit setting definitions for 00, 01 are incorrect; these are combined into “0x”. Also, remove “0.5 cycles” from the Default column. The correct bit setting definition is:

Bit	Default	R/W	Description
9:8	00_2 1.5 cycles	Read/ Write	DRAM Read cycle CAS pulse width (t_{RCP}) - This field affects the number of cycles that CAS7:0# is asserted. Fast Page-Mode DRAM: **0x 1.5 cycles (defaults to 1.5 for FPM DRAM)** 10 2.5 cycles 11 3.5 cycles EDO DRAM and BEDO DRAM (this parameter is fixed for EDO or BEDO DRAM types): xx 0.5 Cycles

24. *Section 14.5.7, Page 14-29, Table 14-12*

The description for Bits 17:16 incorrectly describes the bits as “DRAM Write cycle RAS-to-CAS pulse width (t_{WRC})”. The correct description is “DRAM Write cycle RAS-to-CAS delay (t_{WRC})”.

The description for Bits 9:8 incorrectly describes the bits as “DRAM Write cycle CAS delay (t_{WCP})”. The correct description is “DRAM Write cycle CAS pulse width (t_{WCP})”. Also, the bit setting definitions for 00, 01 are incorrect. The correct bit setting definition follows:

Bit	Default	R/W	Description
9:8	00_2 1.5 cycles	Read/ Write	**DRAM Write cycle CAS pulse width (t_{WCP})** - This field affects the number of cycles that CAS7:0# is asserted. Fast Page-Mode DRAM: **0x 1.5 cycles (defaults to 1.5 for FPM DRAM)** 10 2.5 cycles 11 3.5 cycles EDO DRAM and BEDO DRAM: xx 0.5 Cycles

25. Section 14.5.8, Page 14-31, Table 14-13

In the Bit 16 default, replace “Disabled” with “Enabled”.

26. Section 14.6.2, Page 14-33, Table 14-15

In the Bit 0 Default column, the word “Disabled” is missing.

27. Section 14.7.2, Page 14-38 and 39, Figures 14-17 and 18

Remove # from DALE1:0 signals.

28. Section 15.5.2.1, Page 15-12

Incorrect wording “or equal to” removed from two places in first bullet.

29. Section 15.5.6, Page 15-16

Removed repetitive third paragraph after table 15-3, already fourth paragraph on page 15-15.

30. Section 15.5.6, Page 15-19 and 15-20, Table 15-6

Added “Page 2 of 2” indicator to table title, for page carry-over.

31. Section 15.6.2.1, Page 15-22

In the second paragraph, the last sentence is missing the word “be”. The correct wording is: “The register's upper five bits are programmable; each PCI interface can **be** independently programmed to a value ...“.

32. Section 15.6.2.2, Page 15-23

On Page 15-23, first paragraph, second sentence is missing the word “latched”. The sentence correctly reads: “For writes, the **latched** information includes the data to be written.”

33. Section 15.12.5, Page 15-37

The following sentence is added as the second paragraph in the subsection: “The 80960Rx is a multifunction PCI device. The PCI-to-PCI bridge unit is function zero; the Address Translation Unit is function one.”

34. Section 15.13, Page 15-37

The second sentence in the fourth paragraph reads: “Some registers that are read only from Type 0 Configuration Read and Write commands are writable from the i960 core processor.", the paragraph has been modified to indicate those few registers that are read only from the core. The word “Some” has been removed.

The paragraph now reads: “... Registers that are read only from Type 0 Configuration Read and Write commands are writable from the i960 core processor. As a result, certain configuration registers can be initialized before PCI configuration begins. The i960 core processor reads and writes the bridge configuration space as memory-mapped registers. **Access capabilities from the i960 core processor and from the PCI configuration cycles are as follows:**

- Register bits that are Read/Write from the PCI bus are Read/Write from the i960 core processor
- Register bits that are Read Only from the PCI bus are Read/Write from the i960 core processor
- Register bits that are Read Clear from the PCI bus are Read Clear from the i960 core processor
- Register bits that are Reserved and Read Only from the PCI bus are Reserved and Read Only from the i960 core processor”

35. Section 15.13, Page 15-38, Figure 15-8

The arrows on the right side of the figure incorrectly indicate that the PCI-to-PCI Bridge extends from 00H to 38H, and the i960 Processor-Specific area extends from 3CH to 5CH. The arrow is now “moved down” such that the PCI-to-PCI Bridge extends from 00H to 3CH, and the i960 Processor-Specific area extends from 40H to 5CH.

Also, at offset 34H, the word “**Bridge**” is prepended to the Subsystem ID and Subsystem Vendor ID register names.

36. Section 15.13.4, Page 15-42, Table 15-16

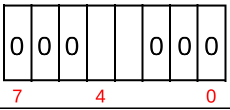
The Bit 08 description, in the first bullet, incorrectly refers to the S_PERR# signal. It now correctly refers to the P_PERR# signal.

37. Section 15.13.7, Page 15-44, Table 15-19

Added fixed bit assignments to Cacheline Size Register for bits 7:5 and 2:0. New Bit definitions follow:

Table 15-19. Cacheline Size Register - CLSR

PCI Configuration Address Offset: 0CH
80960 Core Local Bus Address: 100CH



Bit	Default	R/W	Description
07:05	0H	Read Only	Cacheline Size Maximum - Read only, giving a programmable maximum of 16 for the Cacheline Size.
04:03	00	Read/Write	Cacheline size in DWORDs. Cacheline size is restricted to either 8 or 16 DWORDs.
02:00	0H	Read Only	Cacheline Size Granularity - Read only, giving a programmable granularity of DWORDs for the Cacheline Size.

38. Section 15.13.16, Page 15-50, Table 15-28

Bit 14 was incorrectly defined as: “Signaled SERR# Received”. It is now correctly defined as: “Received SERR#."

39. Section 15.13.23, Page 15-54, Table 15-35

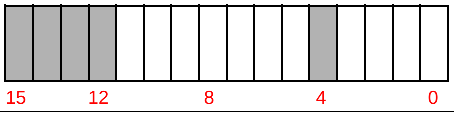
The register’s bit definitions, address offsets, and register diagram are incorrect.

- PCI Configuration Address Offset “36H” is changed to “3EH”
- i960 core processor Local Bus Address “1036H” is changed to “103EH”

The table incorrectly indicates that Bits 15:8 and Bit 4 are Reserved. The correct bit definitions follow. Bold text indicates changed information:

Table 15-35. Bridge Control Register - BCR (Sheet 1 of 4)

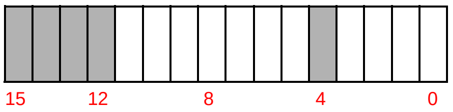
PCI Configuration Address Offset: **3EH**
80960 Core Local Bus Address: **103EH**



Bit	Default	R/W	Description
15:12	0_2	Read Only	Reserved
11	0_2	**Read/ Write**	**Discard Timer SERR# Enable - This bit enables the assertion of SERR# for all discard timers.** **0 - SERR# is not asserted when any discard timer expires.** **1 - SERR# is asserted on the bus where the delayed request was initiated when the discard timer expires.**
10	0_2	**Read Clear**	**Discard Timer Status - This bit indicates the status of the four discard timers.** **0 - no discard timers have expired.** **1 - at least one of the four discard timers has expired.**
09	0_2	**Read/ Write**	**Secondary Discard Timer Value - This bit controls the timeout value for the secondary delayed read and delayed write discard timers.** **0 - the timeout value is 2**15 clocks.** **1 - the timeout value is 2**10 clocks.**
08	0_2	**Read/ Write**	**Primary Discard Timer Value - This bit controls the timeout value for the secondary delayed read and delayed write discard timers.** **0 - the timeout value is 2**15 clocks.** **1 - the timeout value is 2**10 clocks.**
07	0_2	**Read/ Write**	**Fast Back to Back Enable - This bit is ignored.**

Table 15-35. Bridge Control Register - BCR (Sheet 2 of 4)

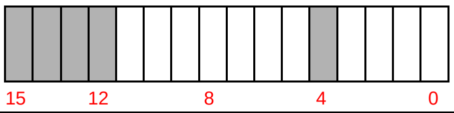
PCI Configuration Address Offset: **3EH**
80960 Core Local Bus Address: **103EH**



Bit	Default	R/W	Description
06	0_2	Read/ Write	Secondary Bus Reset - This bit controls the secondary bus S_RST# signal. When set: • The PCI to PCI Bridge Unit resets all upstream and downstream posting buffers and address queues as well as the secondary PCI bus interface. **The Bridge PCI configuration registers are not reset. The primary PCI bus interface retries all transactions — except Type 0 configuration transactions — until this bit is cleared.** • DMA Channel 2 immediately halts any PCI transactions and returns to an idle state. DMA Channel 2 does not begin any new transfers until the Secondary Bus Reset bit is cleared. • Secondary ATU immediately halts all PCI transactions and completes all local bus transactions. The i960 core processor is released from bus backoff, if necessary. The Secondary ATU does not accept new i960 core processor requests until the Secondary Bus Reset bit is cleared. • **Secondary ATU registers are cleared. These include:** • **Secondary Inbound ATU Base Address Register - SIABAR** • **Secondary Inbound ATU Limit Register - SIALR** • **Secondary Inbound ATU Translate Value Register - SIATVR** • **Secondary Outbound Memory Window Value Register - SOMWVR** • **Secondary Outbound I/O Window Value Register - SOIOWVR** • **Secondary ATU Command Register - SATUCMD** • **Secondary ATU Status Register - SATUSR** • **Secondary Outbound DAC Window Value Register - SODWVR** • **Secondary Outbound Upper 64-Bit DAC Register - SOUDR** • **Secondary Outbound Configuration Cycle Address Register - SOCCAR** When this bit is cleared, the S_TST# signal is deasserted. Software must clear this bit.

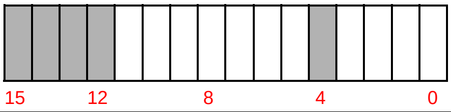
Table 15-35. Bridge Control Register - BCR (Sheet 3 of 4)

PCI Configuration Address Offset: **3EH**
80960 Core Local Bus Address: **103EH**



Bit	Default	R/W	Description
05	0_2	Read/ Write	Master Abort Mode - This bit controls the PCI-to-PCI bridge when a Master-abort termination occurs on either interface when the bridge is the master. When cleared, reads return all ones and write data is accepted by the bridge and dropped. When set, the bridge signals Target-abort to the requesting master when the corresponding transaction on the other side of the bridge terminates with a Master-abort and the transaction is not yet concluded (reads and non-posted writes). When the bit is set and the transaction on the requesting interface has completed (posted writes) the bridge must assert SERR# on the primary interface (providing the function is enabled).
04	0_2	Read Only	Reserved
03	0_2	Read/ Write	VGA Enable - This bit modifies the response by the bridge to VGA compatible addresses. When set, the bridge positively decodes and forwards the following accesses from the primary to secondary interface: - memory accesses where the range is 0A0000H - 0BFFFFH - I/O accesses where AD9:0 are in the range 3B0H - 3BBH and 3C0H - 3DFH (inclusive of ISA addresses - AD15:10 not decoded). VGA address forwarding is independent of the address ranges defined in the memory base registers and the I/O base register. It is also independent of the ISA Enable bit and the VGA Palette Snoop Enable bit. VGA address forwarding is dependent on the state of the I/O and Memory Enable bits. When cleared, the bridge does not forward any VGA addresses.
02	0_2	Read/ Write	ISA Enable - This bit modifies the bridges response to ISA I/O addresses. This only applies to I/O addresses that are defined by the bridge in IOBR and IOLR and are also in the first 64 Kbytes of PCI address space (0000 0000H - 0000 FFFFH) When set, the bridge does not forward from primary to secondary and I/O transactions addressing the upper 768 bytes in each 1 Kbyte block. In the opposite direction, I/O transactions are forwarded up the bridge when the address falls within the upper 768 bytes in each 1 Kbyte block.
01	0_2	Read/ Write	SERR# Enable - This bit controls the forwarding of secondary interface S_SERR# assertions to the primary interface. When the SERR# Enable bit in the PCMDR register is set and the bridge detects the assertion of S_SERR# on the secondary bus, it then asserts P_SERR# on the primary interface.

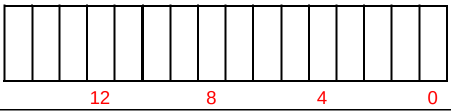
Table 15-35. Bridge Control Register - BCR (Sheet 4 of 4)

PCI Configuration Address Offset: **3EH** 80960 Core Local Bus Address: **103EH**			
Bit	**Default**	**R/W**	**Description**
00	0_2	Read/ Write	Parity Error Response Enable - This bit controls the response to parity errors on the secondary interface. When clear, all address and data parity errors on the secondary interface are ignored. When set, detection and reporting of all parity errors on the secondary interface is enabled. Correct parity must be generated even when parity error reporting is disabled.

40. Section 15.13.23, Pages 15-54, Table 15-34

The top of the table is missing the PCI Configuration Address Offset (36H) and 80960 Core Local Bus Address (1036H), and the bit figure. Bold text indicates changes:

Table 15-34. Bridge Subsystem ID Register - BSIR

PCI Configuration Address Offset: **36H** 80960 Core Local Bus Address: **1036H**			
Bit	**Default**	**R/W**	**Description**
15:0	0H	Read Only	Subsystem ID - This register is used to uniquely identify the add-in board or subsystem

41. Section 16.1, Page 16-2

In the third paragraph (after the bullets), second sentence, “Function 1” is added for clarification. The sentence now reads: “Collectively, these units are the second PCI function **(Function 1)** in the multifunction i960 RP processor."

42. Section 16.2.1, Page 16-5, Figure 16-2

The tag and arrow “Address is not claimed” is added to the top portion of Figure 16-2, above the Base_Register.

43. Section 16.2.1, Page 16-5

In the paragraph that precedes Equation 16-2, the acronym IAQ is introduced yet it has not been defined. The sentence now includes “Inbound Address Queue”.

In the paragraph following Equation 16.2, the last sentence is missing the word “limit”. The sentence now reads: “Address aliasing of multiple PCI addresses to the same physical 80960Rx address can be prevented by programming the inbound **limit** register on boundaries matching the associated limit register, but this only enforced through application programming.”

44. Section 16.2.2, Page 16-7

In the second bullet, remove “inbound address queue”, add the words “the ATU” for three entries, replace the word “read” with “write”. The bullet now reads:

- When no transaction is currently in the IAQ or inbound data queue (IDQ), **the ATU** latch**es** the PCI address into the IAQ. When an inbound write transaction is currently in progress, **the ATU** do**es** not latch the PCI address and signal**s** a Retry to the initiator. When an outbound read transaction is in progress and an inbound write occurs, **the ATU** return**s** any read data to the 80960 local bus. This clears the IDQ, which allows the inbound **write** to occur.”

In the last bullet, second sub-bullet, add an “s” to the word clear and also add “and IDQ” to the end. The sub-bullet now reads: “... clear**s** the IAQ **and IDQ**.”

45. Section 16.2.2, Page 16-8

The two sub-bullets at top of page have changed; bold text indicates changes:

— The IDQ becomes empty **while the transaction on the PCI bus is in progress, but held in wait states**. In this case, the local bus interface goes idle **and is requested again when data is received in the IDQ**.

— The IDQ becomes empty **and the PCI transaction has completed**. The IAQ is cleared, **in this case,** and the local bus interface goes idle. The IAQ and IDQ are now ready for a new transaction.

46. Section 16.2.3, Page 16-8

Added second paragraph which reads:

“Data for all inbound ATU read transactions is implicitly prefetchable as defined in the *PCI Local Bus Specification*, revision 2.1. The Inbound ATU Base Address Register’s Bit 3 is hardwired to one (1) defining the memory space as prefetchable. The ATU prefetches on both single- and multi-word read transactions.”

47. Section 16.2.6.1, Page 16-10

The second paragraph is modified. Bold text indicates changes:

"Figure 16-5 illustrates the outbound address translation windows. Each ATU has **three** windows. **Two are** 64 Mbyte and one is 64 Kbyte. The primary outbound memory and DAC translation windows range from 8000 0000H to 87FF FFFFH **(2 x 64 Mbyte)** and the secondary outbound memory and DAC translation windows range from 8800 0000H to 8FFF FFFFH **(2 x 64 Mbyte)**. After these four windows, the primary and secondary outbound I/O windows range from 9000 0000H to 9001 FFFFH **(2 x 64 Kbyte)**."

48. Section 16.2.6.1, Page 16-11, Figure 16-4

Start of Peripheral Memory Mapped Registers incorrectly stated as "0000 0800H" — this has been replaced with "0000 1000H".

End of Peripheral Memory Mapped Registers incorrectly stated as "0000 1000H" — this has been replaced with "0000 1800H".

49. Section 16.2.6.1, Page 16-12

In the first paragraph at the top of page 16-12, the second sentence is missing the words "Outbound" and "DAC". In the third sentence, replace "have" with "use". The sentence now reads:

"The **Outbound** Upper 64-bit **DAC** registers are for DAC commands and contain the high-order 32 bits of a dual-cycle 64-bit address directly with no translation. Both ATUs **use** the following registers..."

50. Section 16.2.6.1, Page 16-13, Figure 16-5

Figure 16-5 is missing the word "80960RP" in the figure name. It now reads: "Figure 16-5. Outbound 80960RP Address Translation Windows".

51. Section 16.2.6.2, Page 16-14

The last sentence in the paragraph below Figure 16-6, "ATUCR" is not defined. The sentence now reads: "The following bits within the **Address Translation Unit Configuration Register** (ATUCR) affect direct ..."

52. Section 16.5, Page 16-18, Figure 16-7

The upper left heading is missing "or Secondary". The heading now reads: "Primary **or Secondary** PCI Bus".

53. Section 16.5.1, Page 16-18

In the first line replace “queue contains” with “queues contain”. The sentence now reads: “As indicated in Figure 16-7, the ATU transaction **queues contain** three...”

54. Section 16.6, Page 16-20, Table 16-3

The word “Error” was missing from the error condition for Local Bus Master. The condition now reads: “80960RP Memory Controller Parity **Error**”. Also, references to “80960” should be “80960RP”.

55. Section 16.7, Page 16-24

The second paragraph is rewritten to read: “Figure 16-8 defines the **format for the first 64 bytes of the header. The additional 182 bytes of the configuration space is defined as the ATU Extended configuration space**.”

56. Section 16.7, Page 16-25, Figure 16-8

The shaded area (34H, 38H) should be marked as “Reserved”.

57. Section 16.7, Page 16-26, Table 16-8

Column Heading for the first column should read: “**PCI Configuration Address Offset**”, not “PCI Address”.

The entry for A0H is incorrect; it should read: **“A0H Section 16.7.38, Secondary Outbound Upper 64-bit DAC Register - SOUDR (pg. 16-58)”**

58. Section 16.7.2, Page 16-29, Table 16-11

Added fixed bit assignments to register for 15:0, “0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0”, also the default value incorrectly states “I960H”. Now correctly states “1960H”.

59. Section 16.7.4, page 16-31

Replace the last sentence of the paragraph with: “The *read/clear* bits can only be set by the internal hardware and are cleared by either a reset condition or by writing a 1_2 to the bit to be cleared.”

60. Section 16.7.7, Page 16-33, Table 16-16

Added fixed bit assignments to register for 7:5, “0 0 0” and 2:0, “0 0 0”.

61. Section 16.7.8, Page 16-33, Table 16-17

Added text to end of description for bits 07:03. Description now reads: "... from 0 to 248 clocks, **in increments of eight clocks**."

62. Section 16.7.9, Page 16-34, Table 16-18

Added text to end of description for bits 06:00. Description now reads: "... *PCI Local Bus Specification*, revision 2.1. **Type 00H configuration space header definition**."

63. Section 16.7.11, Page 16-35

The last sentence of the first paragraph is changed to: "When a value of FFFF FFFFH or FFFF FFFEH is written to the PIABAR, the next read from the register returns data from the Primary Inbound ATU Limit Register (PIALR) and not the PIABAR. All subsequent reads of the PIABAR return the contents of the PIABAR."

64. Section 16.7.14.1, Page 16-38

Change the first sentence of the first paragraph to read:

"The required address size and type can be determined by writing **FFFF FFFFH** or **FFFF FFFEH** to a base register, and then reading **back** from the register."

On page 16-39, modify last sentence in the paragraph to read:

The first read after a write of FFFF FFFFH **or FFFF FFFEH is** directed to the Limit Register instead of the base address register. **If any other value is written to a base address register, that value is programmed into the register. The data returned on subsequent reads from the register is the contents of the base address register (with all writable bits programmed) — NOT the contents of the corresponding limit register.**

In Table 16-24, add the following notes:

NOTES:

1. **The minimum size is 2 Kbytes for the Expansion ROM block.**
2. **The minimum size is 4 Kbytes for the ATU Memory block.**

65. Section 16.7.19, Page 16-43, Table 16-30

11:00 Default incorrectly stated at "0000_2" replaced with "000_2".

66. Section 16.7.21, Page 16-44

Second paragraph, second line, change the word “Primary” to “Secondary”. Bold text shows changes:

“The effects on the base address register are that when a value of FFFF FFFFH is written to the SIABAR, the next read from the register returns data from the **Secondary** Inbound ATU Limit Register (SIALR) and not the SIABAR.”

67. Section 16.7.30, Page 16-50

Add these two paragraphs to follow the existing paragraph:

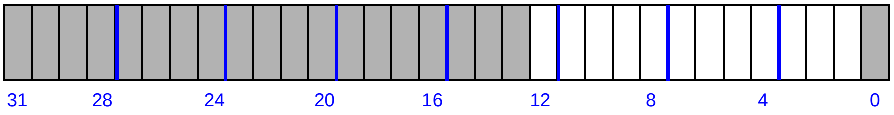
“The Expansion ROM base address is specified in Section 16.7.14, Expansion ROM Base Address Register - ERBAR (pg. 16-37). When determining the block size requirements as described in Section 16.7.14.1, Determining Block Sizes for Base Address Registers (pg. 16-38), the Expansion ROM Limit Register provides the block size requirements for the Expansion ROM Base Address Register.

The programmed value within the 80960RP value register must be naturally aligned with the programmed value found in the base address register. The limit register is used as a mask; thus, the lower address bits programmed into the 80960RP value register are invalid. Refer to the *PCI Local Bus Specification* Revision 2.1 for additional information on programming base address registers.”

68. Section 16.7.32, Page 16-51, Table 16-43

Bit 9 description incorrect and replaced. Bit 10 R/W incorrectly stated as “Read Only” replaced with “Read/Write”. Bit 10 description incorrect and replaced. Bit 12 was previously defined as “Reserved”; it is now defined as “Secondary Bus - Messaging Unit Access Enable” (see also Specification Changes item titled “5., ATU Configuration Register Bit 12 Definition is Changed”. The correct bit definitions are indicated in bold text in the following table.

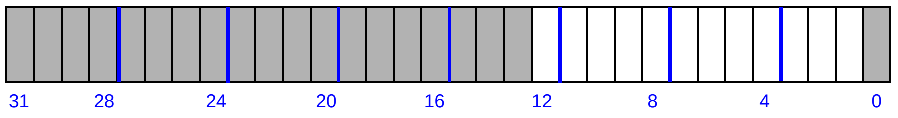
Table 16-43. ATU Configuration Register - ATUCR (Sheet 1 of 2)



PCI Configuration Address Offset: 88H
80960 Core Local Bus Address: 1288H

Bit	Default	R/W	Description
31:13	00000H	Read Only	Reserved
12	**0_2**	**Read/Write**	**Secondary Bus, Messaging Unit Access Enable - When set, the PCI-to-PCI Bridge unit can forward a transaction from the secondary PCI interface, through the bridge, and to the Messaging Unit (first 4 Kbytes of the PATU inbound address space) on the primary PCI interface. For correct operation, the transaction must be a valid bridge address (claimed by secondary interface of the bridge and forwarded to the primary interface of the bridge) as well as a valid Messaging Unit address. When clear, the Messaging Unit cannot claim a transaction mastered by the primary interface of the bridge.**
11	0_2	Read/Write	Secondary PCI Boot Mode - When set, the secondary ATU claims all local bus accesses with addresses in the range FE00 0000H to FFFF FFFFH. This allows the i960 core processor to boot from the secondary PCI bus. The translation algorithm uses the Secondary Outbound I/O Window Value Register in this mode.
10	**0_2**	**Read/Write**	**Secondary SERR Interrupt Enable - When set, the i960 core processor receives an NMI# when the Primary ATU detects that S_SERR# was asserted. When clear, no interrupt is sent.**
09	**0_2**	**Read/Write**	**Primary SERR Interrupt Enable - When set, the i960 core processor receives an NMI# when the Primary ATU detects that S_SERR# was asserted. When clear, no interrupt is sent.**
08	0_2	Read/Write	Direct Addressing Enable - When set, enables direct addressing through the ATUs. Local bus cycles with an address between 0000 1000H and 7FFF FFFFH are automatically forwarded to the PCI bus with no address translation. The ATU which claims the direct addressing transaction depends on the Secondary Direct Addressing Select bit state.

Table 16-43. ATU Configuration Register - ATUCR (Sheet 2 of 2)



PCI Configuration Address Offset: 88H
80960 Core Local Bus Address: 1288H

Bit	Default	R/W	Description
07	0_2	Read/Write	Secondary Direct Addressing Select - When set, results in direct addressing outbound transactions to be forwarded through the secondary ATU to the secondary PCI bus. When clear, direct addressing uses the primary ATU and the primary PCI bus. The Direct Addressing Enable bit must be set to enable direct addressing.
06	0_2	Read/Write	Expansion ROM Width - When clear, this bit signifies that an 8-bit Expansion ROM is being used. When set, this bit signifies that 32-bit Expansion ROM is in use. Used in conjunction with the ERBAR address decode enable (bit 0).
05	0_2	Read/Write	Secondary ATU PCI Error Interrupt Enable - This bit acts as a mask for Secondary ATU Interrupt Status Register bits 4:0. When set, enables an interrupt to the i960 core processor when any of these bits are set in the SATUISR. When cleared, disables the interrupt.
04	0_2	Read/Write	Primary ATU PCI Error Interrupt Enable - This bit acts as a mask for Primary ATU Interrupt Status Register bits 4:0. When set, enables an interrupt to the i960 core processor when any of these bits are set in the PATUISR. When cleared, disables the interrupt.
03	0_2	Read/Write	ATU BIST Interrupt Enable - When set, enables an interrupt to the i960 core processor when the start BIST bit is set in the ATUBISTR register. This bit is also reflected as the BIST Capable bit 7 in the ATUBISTR register.
02	0_2	Read/Write	Secondary Outbound ATU Enable - When set, enables the secondary outbound address translation unit. When cleared, disables the secondary outbound ATU.
01	0_2	Read/Write	Primary Outbound ATU Enable - When set, enables the primary outbound address translation unit. When cleared, disables the primary outbound ATU.
00	0_2	Read Only	Reserved

69. Section 16.7.33, page 16-53, Table 16-44

The Read/Write definition for all bits in the table are changed to Read Only. The first paragraph in this section correctly states that all bits in this register are Read Only from PCI and Read/Clear from the local bus.

Also, in the description for bit 00, PCI Master Parity Error: remove the last sentence that reads, “Bit is cleared by the host processor ...”

70. Section 16.7.33, Page 16-54, Table 16-44

The bit 02 description should read: "PCI Target Abort (master) - set when a transaction initiated by the ATU master interface ends in a Target-abort."

71. Section 16.7.34, page 16-54, Table 16-45

The Read/Write definition for all bits in the table are changed to Read Only. The first paragraph in this section correctly states that all bits in this register are Read Only from PCI and Read/Clear from the local bus.

Replace P_SERR# with S_SERR# in the description for bit 4.

Also, in the description for bit 00, PCI Master Parity Error: remove the last sentence that reads, “Bit is cleared by the host processor ...”

72. Section 16.7.34, Page 16-55, Table 16-45

The bit 02 description should read: "PCI Target Abort (master) - set when a transaction initiated by the ATU master interface ends in a Target-abort."

73. Section 16.7.35, Page 16-56, Table 16-46

Incorrectly shows the 16-bit Secondary ATU Command Register as a 32-bit register. The Figure at top of table is changed to display 16 bits (15:00), and the first table entry is:

Bit	Default	R/W	Description
15:10	00H	Read Only	Reserved

74. Section 16.7.36, page 16-57

The paragraph is modified to read: “Secondary ATU Status Register bit definitions adhere to PCI Local Bus Specification Revision 2.1 for configuration space device status. The read/clear bits can only be set by the internal hardware and are cleared by either a reset condition or by writing a 12 to the bit to be cleared.”

75. Section 16.7.37, page 16-57, Table 16-47

Replace table contents with the following:

Bit	Default	R/W	Description
15	0_2	Read/Clear	Parity Error - set when a parity error is detected on the secondary PCI bus even when the SATUCMD Register's Parity Checking Enable bit is clear. 0 - S_SERR# no asserted 1 - S_SERR# asserted
14	0_2	Read/Clear	S_SERR# Asserted - set when the Secondary ATU asserted S_SERR# on the PCI bus. 0 - no parity error detected 1 - parity error detected
13:00	0000H	Read Only	Reserved

76. Section 16.7.39, Page 16-58

Add the following paragraph after the first paragraph in this section:

"The value programmed into these registers is not a byte address. See the *PCI Local Bus Specification* Revision 2.1 for information regarding configuration address cycle formats."

77. Section 16.7.40, Page 16-59

Add the following paragraph after the first paragraph in this section:

"The value programmed into these registers is not a byte address. See the *PCI Local Bus Specification* Revision 2.1 for information regarding configuration address cycle formats."

78. Section 16.7.41, Page 16-59 and 60

Add the following paragraph after the first paragraph in this section:

"The configuration cycle generated on the PCI bus enables the same bytes which are accessed in the corresponding data register. For example, a read of all 32 bits of this data register generates a 4-byte configuration read cycle on the Primary PCI Bus of the addressed configuration register. Also, a write of byte 2 (bits 23:16) of this data register generates a single byte configuration write cycle of byte 2 of the addressed configuration register. Similar actions take place for short accesses."

On page 60, in the first paragraph following the bullets: change ACH to 12ACH. Bold indicates the corrected text:

"...The 80960 Core Local Bus address is **12ACH**."

79. Section 16.7.42, Page 16-60

Add the following paragraph after the first paragraph in this section:
"The configuration cycle generated on the PCI bus enables the same bytes which are accessed in the corresponding data register. For example, a read of all 32 bits of this data register generates a 4-byte configuration read cycle on the Secondary PCI Bus of the addressed configuration register. Also, a write of byte 2 (bits 23:16) of this data register generates a single byte configuration write cycle of byte 2 of the addressed configuration register. Similar actions take place for short accesses."

In the first paragraph following the bullets: change B0H to 12B0H. Bold indicates the corrected text:

"...The 80960 Core Local Bus address is **12B0H**."

80. Section 17.7.20, Page 17-30, Figure 17-3

Added "Entries" to table headings. Replaced all table values. The table now reads:

Queue Offset Registers	**4 Kbyte Entries Qsize**	**8 Kbyte Entries Qsize**	**16 Kbyte Entries Qsize**	**32 Kbyte Entries Qsize**	**64 Kbyte Entries Qsize**
IFHPR IFTPR	0	0	0	0	0
IPHPR IPTPR	4000H	8000H	10000H	20000H	40000H
OPHPR OPTPR	8000H	10000H	20000H	40000H	80000H
OFHPR OFTPR	C000H	18000H	30000H	60000H	C0000H

81. Section 18.2.3, Page 18-6

In the first sentence, added "when the latency timer times out". The sentence now reads:
"... of the local bus**, when the latency timer times out,** under the following conditions:"

82. Section 18.3.2, Page 18-11

The first sentence of the second paragraph: "Enable" is changed to "Status". Bold text indicates the changes:

The SACR register also contains the Secondary Arbiter **Status** bit for the secondary bus arbitration unit. This bit is set at reset by sampling the S_REQ5#/S_ARB_EN signal. When this bit is clear, the secondary bus arbiter is disabled and the bridge drives S_REQ# on S_GNT0# and samples S_GNT# on S_REQ0#. When S_REQ5#/S_ARB_EN is high on the rising edge of P_RST#, the internal secondary arbitration unit is enabled. When S_REQ5#/S_ARB_EN is low on the rising edge of P_RST#, the internal secondary arbitration unit is disabled.

83. Section 18.3.2, Page 18-11, Table 18-9

Bit 16 definition has changed. Bold text indicates changes:

Bit	Default	R/W	Description
16	Based on S_REQ5#/S_ARB_EN signal at reset	**Read Only**[1]	Secondary Arbiter **Status** 0 = Disabled 1 = Enabled

NOTES:

1. **This bit is Read Only from BOTH the PCI interface and the 80960Rx local bus. This is unlike other Read Only bits which are read only from the PCI interface, and Read/Write from the 80960Rx local bus.**

84. Section 20.4, Page 20-11

The fourth paragraph discusses PCI master and slave devices inserting wait states. This topic is not dependent on demand mode DMA. In the next release of the user's manual, this paragraph will become a new section titled: "**20.5 WAIT STATES INITIATED BY PCI DEVICES**".

85. Section 20.4, Page 20-17, Figure 20-12

The asterisk (*) and arrow should refer to the rising edge of the DREQ# signal, not the rising edge of the DACK# signal.

The note text changes as follows:

NOTE: * Asterisk indicates the device ENDs transfer by deasserting DREQ#.

86. Section 20.4, Page 20-18, Figure 20-13

The asterisk (*) and arrow should refer to the rising edge of the DREQ# signal, not the rising edge of the DACK# signal.

The note text changes as follows:

NOTE: * Asterisk indicates the device ENDs transfer by deasserting DREQ#.

87. Section 20.5.1, Page 20-19

In the fourth bullet (last bullet on page), from the second-to-last sentence, remove "...**while accessing the Memory Controller**..." It should now read:

"When the DMA channel reaches a 2-Kbyte address boundary, the DMA controller stops the current local bus transaction."

In the same bullet, remove the last sentence:

The DMA channel does not implement the 2-Kbyte address boundary when accessing memory controllers that are external to the i960 RP processor.

88. Section 20.5.2, Page 20-20

In the second bullet (preceding the last bullet on page), from the second-to-last sentence, remove "...**while accessing the Memory Controller**..." It should now read:

"When the DMA channel reaches a 2-Kbyte address boundary, the DMA controller stops the current local bus transaction."

In the same bullet, remove the last sentence:

The DMA channel does not implement the 2-Kbyte address boundary when accessing memory controllers that are external to the i960 RP processor.

89. Section 22.2, Page 22-1

Remove the first two sentences of the last paragraph on the page (they are redundant from the previous paragraph).

90. Section 23.2.4.4, Page 23-9, Table 23-4

The Boundary Scan Register Bit Order shown in Table 23-4 is incorrect (out-of-date). The correct bit order follows:

- Length of RP_Processor is 264
- The first cell, cell 0, is closest to TDO

Bit#	Signal	Bit#	Signal	Bit#	Signal	Bit#	Signal
0	BLAST#	1	DEN#	2	DT/R#	3	W/R#
4	BE#(0)	5	BE#(1)	6	control	7	control
8	control	9	BE#(2)	10	BE#(3)	11	ADS#
12	ALE	13	LRDYRCV#	14	RDYRCV#	15	control
16	AD(0)	17	AD(1)	18	AD(2)	19	AD(3)

Bit#	Signal	Bit#	Signal	Bit#	Signal	Bit#	Signal
20	AD(4)	21	AD(5)	22	AD(6)	23	AD(7)
24	AD(8)	25	AD(9)	26	AD(10)	27	AD(11)
28	AD(12)	29	AD(13)	30	AD(14)	31	AD(15)
32	control	33	AD(16)	34	AD(17)	35	AD(18)
36	AD(19)	37	AD(20)	38	AD(21)	39	AD(22)
40	AD(23)	41	AD(24)	42	AD(25)	43	AD(26)
44	AD(27)	45	AD(28)	46	AD(29)	47	AD(30)
48	AD(31)	49	control	50	MA(0)	51	MA(1)
52	MA(2)	53	MA(3)	54	MA(4)	55	MA(5)
56	MA(6)	57	MA(7)	58	MA(8)	59	MA(9)
60	MA(10)	61	MA(11)	62	control	63	DP(0)
64	DP(1)	65	DP(2)	66	DP(3)	67	RAS#(0)
68	RAS#(1)	69	RAS#(2)	70	RAS#(3)	71	CAS#(0)
72	CAS#(1)	73	CAS#(2)	74	CAS#(3)	75	CAS#(4)
76	CAS#(5)	77	CAS#(6)	78	CAS#(7)	79	MWE#(0)
80	MWE#(1)	81	MWE#(2)	82	MWE#(3)	83	DWE#(0)
84	DWE#(1)	85	CE#(0)	86	CE#(1)	87	LEAF#(0)
88	LEAF#(1)	89	DALE(0)	90	DALE(1)	91	WAIT#
92	S_INTA/ XINT#0	93	S_INTB/ XINT#1	94	S_INTC/ XINT#2	95	S_INTD/ XINT#3
96	control	97	XINT#4	98	XINT#5	99	XINT#6
100	XINT#7	101	NMI#	102	control	103	control
104	control	105	control	106	PICD(0)	107	PICD(1)
108	PICCLK	109	SCL	110	SDA	111	HOLDA
112	HOLD	113	DACK#	114	DREQ#	115	STEST
116	LOCK#/ ONCE#	117	D/C#/ RST_MODE#	118	FAIL#	119	WIDTH/ HLTD0/ SYNC
120	WIDTH/ HLTD1/ RETRY	121	LRST#	122	control	123	control
124	P_INTA#	125	P_INTB#	126	P_INTC#	127	P_INTD#
128	P_RST#	129	P_CLK	130	P_GNT#	131	P_REQ#
132	control	133	control	134	control	135	control
136	control	137	P_AD(31)	138	P_AD(30)	139	P_AD(29)
140	P_AD(28)	141	P_AD(27)	142	P_AD(26)	143	P_AD(25)
144	P_AD(24)	145	P_C/BE#(3)	146	control	147	P_IDSEL
148	P_AD(23)	149	P_AD(22)	150	P_AD(21)	151	P_AD(20)

Bit#	Signal	Bit#	Signal	Bit#	Signal	Bit#	Signal
152	P_AD(19)	153	P_AD(18)	154	P_AD(17)	155	P_AD(16)
156	control	157	control	158	P_C/BE#(2)	159	P_FRAME#
160	control	161	control	162	P_IRDY#	163	P_TRDY#
164	P_DEVSEL#	165	P_STOP#	166	P_LOCK#	167	P_PERR#
168	P_SERR#	169	P_PAR	170	P_C/BE#(1)	171	control
172	control	173	control	174	P_AD(15)	175	P_AD(14)
176	P_AD(13)	177	P_AD(12)	178	P_AD(11)	179	P_AD(10)
180	P_AD(9)	181	P_AD(8)	182	control	183	P_C/BE#(0)
184	P_AD(7)	185	P_AD(6)	186	P_AD(5)	187	P_AD(4)
188	P_AD(3)	189	P_AD(2)	190	P_AD(1)	191	P_AD(0)
192	S_AD(0)	193	S_AD(1)	194	S_AD(2)	195	S_AD(3)
196	S_AD(4)	197	S_AD(5)	198	S_AD(6)	199	S_AD(7)
200	control	201	S_C/BE#(0)	202	S_AD(8)	203	S_AD(9)
204	S_AD(10)	205	S_AD(11)	206	S_AD(12)	207	S_AD(13)
208	S_AD(14)	209	S_AD(15)	210	S_C/BE#(1)	211	S_PAR
212	S_SERR#	213	control	214	control	215	control
216	S_PERR#	217	S_LOCK#	218	S_STOP#	219	S_DEVSEL#
220	S_TRDY#	221	S_IRDY#	222	S_FRAME#	223	S_C/BE#(2)
224	control	225	control	226	control	227	control
228	control	229	S_AD(16)	230	S_AD(17)	231	S_AD(18)
232	S_AD(19)	233	S_AD(20)	234	S_AD(21)	235	S_AD(22)
236	S_AD(23)	237	control	238	S_IDSEL	239	S_C/BE#(3)
240	S_AD(24)	241	S_AD(25)	242	S_AD(26)	243	S_AD(27)
244	S_AD(28)	245	S_AD(29)	246	S_AD(30)	247	S_AD(31)
248	S_RST#	249	S_REQ0/ S_GNT#	250	S_GNT0/ S_REQ#	251	S_REQ#(0)
252	S_GNT#(0)	253	S_REQ#(1)	254	S_GNT#(1)	255	S_REQ#(2)
256	S_GNT#(2)	257	control	258	control	259	S_CLK
260	S_REQ#(3)	261	S_GNT#(3)	262	S_REQ#(4)	263	S_GNT#(4)

91. Appendix A, Page A-4, Table A-5

Under src2, option 1, replace “sf0...sf4” with “reserved”.