i960® RP/RD I/O PROCESSOR SPECIFICATION UPDATE

Release Date: June, 1998
Order Number: 272918-019

The i960® RP/RD I/O Processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
PO Box 5937
Denver CO 80217-9808

or call 1-800-548-4725

or visit Intel's website at http://www.intel.com

REVISION HISTORY

Date	Revision	Description
06/10/98	019	Added documentation changes 13 and 14.
05/14/98	018	Added errata 31 and 32. Added Specification changes 12 through 20.
01/24/98	017	Added errata 29 and 30.
01/12/98	016	Changed all references to “80960Rx” to “80960RP/RD”.
12/05/97	015	Added documentation changes 10, 11, 12. Updated Table 10-1 in this spec update to reflect the data in the *i960® Rx I/O Microprocessor Developer's Manual* (272736-002). Previous changes in Table 10-1 of this spec update were validto *i960® RP Microprocessor User's Manual* (272736-001).
11/12/97	014	Added Documentation Changes for *i960® Rx I/O Microprocessor Developer's Manual* (272736-002), items 1, 2, 3, 4, 5, 6, 7, 8, 9. Removed all Documentation Changes to previous revision of Developer's Manual (272736-001). See SUMMARY TABLE OF CHANGES.
10/15/97	013	Added errata 27 and 28.
9/12/97	012	Added additional Topside Markings information.
8/13/97	011	Added Topside Markings information.
7/09/97	010	Added errata 25, 26 and specification clarification 9.
5/14/97	009	The Implication of errata item 23 was clarified. Added Documentation Changes, see SUMMARY TABLE OF CHANGES.
4/15/97	008	Added errata items 23 and 24. In SUMMARY TABLE OF CHANGES Errata table, added column for 80960RP/RD 33/66/3.3 B0 step.
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 in 272736-001. Added new stepping information; see Topside Markings. 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.

REVISION HISTORY

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 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® Rx I/O Microprocessor Developer's Manual	272736-002
i960® RP I/O Processor at 5 Volts Data Sheet	272737-003
i960® RP/RD I/O Processor at 3.3 Volts Data Sheet	273001-002

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.

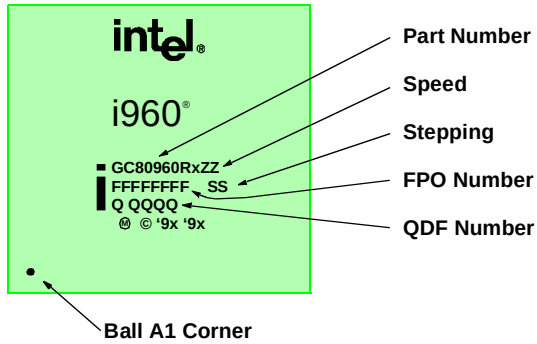
Documentation Changes include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

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.).

IDENTIFICATION INFORMATION

Topside Markings



Part Number[1]	Stepping	QDF Number	Voltage (V)	i960 Core Processor Speed (MHz)	Notes
GC80960RP33	A-0	Q 8163	5.0	33	Samples - limited testing
GC80960RP33	A-1	Q 896	5.0	33	Samples - limited testing
GC80960RP33	A-1	Q 8255	5.0	33	Samples - limited testing
GC80960RP33	A-1	none	5.0	33	Production
GC80960RP33	A-0	Q 8334	3.3	33	Samples - limited testing
GC80960RD66	A-0	Q 8333	3.3	66	Samples - limited testing
GC80960RP3V33	B-0	Q 8419	3.3	33	Samples - limited testing
GC80960RP3V33	B-0	Q 8420	3.3	33	Samples - full production testing
GC80960RP3V33	B-0	none	3.3	33	Production
GC80960RD66	B-0	Q 8421	3.3	66	Samples - limited testing
GC80960RD66	B-0	Q 8422	3.3	66	Samples - full production testing
GC80960RD66	B-0	none	3.3	66	Production

NOTES:

1. GC = 352 HL-PBGA (SuperBGA* package)

Device ID Registers

80960RP/RD processors may be identified electrically according to device type and stepping. Refer to the *i960® Rx I/O Microprocessor Developer's Manual* for bit descriptions of each identification register.

Device and Stepping	Processor Device ID Register (PDIDR - 1710H) (g0)	PCI-to-PCI Bridge Unit Revision ID Register (RIDR - 1008H)	Address Translation Unit Revision ID Register (ATURID - 1208H)	i960 Core Processor Device ID (DEVICEID - FF00 8710H)
80960RP 33/5.0 A-0	0x00860013	0x00	0x00	0x28820013
80960RP 33/5.0 A-1	0x10860013	0x01	0x01	0x28820013
80960RP 33/3.3 A-0 80960RD 66/3.3 A-0	0x08861013 0x08861013	0x02 0x02	0x02 0x02	0x28820013 0x28830013
80960RP 33/3.3 B-0 80960RD 66/3.3 B-0	0x18861013 0x18861013	0x03 0x03	0x03 0x03	0x28820013 0x28830013

SUMMARY TABLE OF CHANGES

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the 80960RP/RD 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		80960RP 33/3.3 80960RD 66/3.3				
	A-0	A-1	A-0	B-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	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 simultaneously 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
14	X	X			20	Fix	Access to non-existent DRAM causes incorrect error reporting
15	X	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

Errata (Sheet 2 of 2)

Item	Stepping				Page	Status	Errata
	80960RP 33/5.0		80960RP 33/3.3 80960RD 66/3.3				
	A-0	A-1	A-0	B-0			
17	X	X			22	Fix	Messaging Unit logs multiple interrupt sources when using the index registers
18	X	X	X	X	24	NoFix	DMA Descriptors appended to the end of a chain may not execute
19	X	X	X	X	23	NoFix	Downstream exclusive read transaction through multiple bridges may cause a deadlock condition
20	X	X	X		24	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			25	Fix	PCI-to-PCI Bridge Unit can corrupt data during Memory Write cycles when wait states are inserted simulta-neously on both the primary and secondary sides of the bridge
23	X	X	X	X	26	Eval	Memory Controller Unit may assert an unexpected RAS# in certain memory configurations
24	X	X	X		30	Fix	ATU may deadlock when the PCI delayed read discard timer expires just as the delayed read completion cycle begins
25				X	30	Eval	Changing the limit register when a value exists in the corresponding base address register may prevent access to the address space
26	X	X	X	X	32	Eval	Messaging Unit may drop messages during simulta-neous inbound primary PCI bus reads and 80960 local bus reads
27	X	X	X	X	33	Eval	P_REQ# is not deasserted when a single DWORD transfer is retried
28		X			33	Fix	Bit 0 of the ATURID register is permanently set to a one
29	X	X	X	X	34	NoFix	VGA memory addresses are aliased throughout the PCI memory address range by the PCI-to-PCI Bridge when the VGA Enable bit is set.
30	X	X	X	X	34	NoFix	PCI memory addresses that are aliases of VGA I/O addresses will not be forwarded from the secondary to the primary PCI Bus interface if the VGA Enable bit is set.
31	X	X	X	X	35	NoFix	Inbound ATU writes to non-existent 80960 local memory will cause the next PCI configuration write cycle to target abort on the PCI bus.
32	X	X	X	X	36	NoFix	Inbound configuration write cycles may latch invalid data on the PCI bus if STOP# is asserted before the initiator asserts IRDY# during the delayed request cycle.

Specification Changes

Item	Stepping				Page	Status	Specification Changes
	80960RP 33/5.0		80960RP 33/3.3 80960RD 66/3.3				
	A-0	A-1	A-0	B-0			
1	X	X			37	Doc	PCI Interrupt Routing Select Register (PIRSR) Polarity
2	X	X	X	X	37	Doc	Asynchronous clocking mode not supported
3	X	X	X	X	37	Doc	The MU interlock mechanism remains enabled for APIC registers when the APIC is disabled
4		X	X	X	38	Doc	Incorrect BIOS access of block size from Base Address Registers
5	X	X	X	X	38	Doc	ATU Configuration Register Bit 12 Definition is Changed
6					39	Doc	Multiple reads of the Base Address Register after writing all 1's will return different values (See Specification Clarification Item #5)
7			X	X	39	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	X	39	Doc	PCI Interrupt Routing Select Register (PIRSR) supports individual routing of each XINT3:0# pin
9			X	X	39	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	X	40	Doc	Mask bits added for all PCI-PCI bridge error conditions which may cause an NMI# to the i960 core processor
11			X	X	42	Doc	Multiple reads of the Base Address Register, after writing all 1's, returns the limit register value until rewritten
12	X	X	X	X	42	Doc	Burst EDO (BEDO) memories are no longer supported
13	X	X	X	X	42	Doc	The I/O APIC is no longer supported
14	X	X	X	X	43	Doc	The PCI-to-PCI bridge no longer supports secondary positive memory decoding
15	X	X	X	X	43	Doc	The PCI-to-PCI bridge no longer supports secondary positive I/O decoding
16	X	X	X	X	43	Doc	Secondary PCI Boot Mode
17	X	X	X	X	43	Doc	32-bit PCI expansion ROM support
18	X	X	X	X	43	Doc	64-bit Dual Address Cycles (DACs) are not supported
19	X	X	X	X	44	Doc	The forwarding of configuration writes from the secondary PCI bus through the PCI-to-PCI bridge is not supported
20	X	X	X	X	44	Doc	Messaging Unit Index Registers are no longer supported

Specification Clarifications

Item	Stepping				Page	Status	Specification Clarifications
	80960RP 33/5.0		80960RP 33/3.3 80960RD 66/3.3				
	A-0	A-1	A-0	B-0			
1	X	X			45	Doc	Five IDSEL lines for the secondary PCI bus are not enabled
2	X	X	X	X	45	Doc	How to use the Data Enable (DEN#) Signal with an In-Circuit Emulator
3	X	X	X	X	46	Doc	Accessing 80960RP/RD MMRs via the primary PCI bus
4	X	X	X	X	47	Doc	Parity on data cycles not checked before delayed write completion cycles are accepted
5	X	X			47	Doc	Multiple reads of the Base Address Register after writing all 1's will return different values
6	X	X	X		47	Doc	When determining memory address block size, accesses to the Base Address Register must be 32-bit configuration cycles
7	X	X	X	X	48	Doc	Secondary PCI bus reset notification to i960 core processor
8	X	X	X	X	49	Doc	Parity error reporting during configuration cycles
9	X	X	X	X	49	Doc	Determining Block Sizes for Base Address Registers

Documentation Changes

Item	Document Revision	Page	Documentation Changes
1	272736-002	53	Section 8.3.2, Page 8-26
2	272736-002	55	Section 8.4.1, Page 8-32, Table 8-8
3	272736-002	56	Section 8.4.1, Page 8-32, Table 8-9
4	272736-002	56	Section 11.7.4, Page 11-27
5	272736-002	57	Section 11.7.4, Page 11-27, Figure 11-6
6	272736-002	58	Section 12.3.1, Page 12-6, Table 12-3
7	272736-002	59	Section 14.4, Page 14-5, Figure 14-3
8	272736-002	60	Section 14.6.7, Page 14-31, Table 14-20
9	272736-002	60	Section 16.7.12, Page 16-39
10	272736-002	60	Section 16.7.21, Page 16-47, Table 16-31
11	272736-002	60	Section 15.13.34, Page 15-75, Table 15-45
12	272736-002	61	Section 17.7.11, Page 17-26, Table 17-15
13	272736-002	61	Section 16.7.18, Page 16-44, Table 16-28
14	272736-002	62	Section 16.7.19, Page 16-45, Table 16-29

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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 80960RP/RD 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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 80960 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 80960RP/RD. The LBALCR register should also be programmed to a value greater than 10.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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

PROBLEM: The 80960RP/RD latches write data into the Bridge Control Register (BCR) based on the state of lower byte enable only. The 80960RP/RD does not check the state of upper byte enable. This erratum applies to the configuration writes from the primary PCI interface as well as 80960 local bus writes from the i960 core processor.

IMPLICATION: A byte write to the lower byte of the BCR 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 BCR does not change the contents of the register.

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

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 80960 local memory through the ATU's must generate address parity.

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

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 80960RP/RD 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 80960RP/RD software. This allows software to setup the 80960RP/RD registers before the system host BIOS can configure it. The 80960RP/RD may enter a lockup condition which can only be cleared by a RESET when:

- the first configuration cycle to the 80960RP/RD 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 80960RP/RD is a write cycle, the 80960RP/RD 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 80960RP/RD could be a write cycle, use initialization modes 0 or 2.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 80960 local memory.
3. Don't perform simultaneous inbound reads and outbound writes through the secondary ATU.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 80960 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/RD 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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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, located at local bus offset 1534H.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 80960RP/RD 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 80960RP/RD local memory if a user application depended on the 80960RP/RD 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

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 flag is cleared, set the Chain Resume bit once more.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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

PROBLEM: A deadlock condition is possible with an behind another 80960RP/RD 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 80960RP/RD bridge unit may deadlock.

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

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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 80960RP/RD application does not allow two PCI masters accessing the same memory address location simultaneously through either the primary or secondary ATU.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

23. Memory Controller Unit may assert an unexpected RAS# in certain memory configurations

PROBLEM: The 80960RP/RD MCU supports from one to four banks of DRAM. When the memory subsystem contains fewer than the maximum number of banks used in the 80960RP/RD design, certain addresses may cause the MCU to assert a RAS# to an empty bank. Table 1 shows the relationship between the 80960 local memory address and the RAS# asserted by the MCU.

Table 1. DRAM Bank/Leaf Size and RAS# Asserted

	Non-Interleaved DRAM		Interleaved DRAM		DRAM Base Address Register (DBAR) Address Boundary
DRAM Bank Control Register (DBCR) Bits 2:1	Address Bits	RAS# Signal Asserted	Address Bits	RAS# Signal Asserted	4 * Bank/Leaf Size
00 (1 Mbyte DRAM per bank/leaf)	21:20		21:20		40 0000H (4 Mbytes)
	0 0	RAS0#	00, 01	RAS1:0#	
	0 1	RAS1#			
	1 0	RAS2#	10, 11	RAS3:2#	
	1 1	RAS3#			
01 (4 Mbyte DRAM per bank/leaf)	23:22		23:22		100 0000H (16 Mbytes)
	0 0	RAS0#	00, 01	RAS1:0#	
	0 1	RAS1#			
	1 0	RAS2#	10, 11	RAS3:2#	
	1 1	RAS3#			
10 (16 Mbyte DRAM per bank/leaf)	25:24		25:24		400 0000H (64 Mbytes)
	0 0	RAS0#	00, 01	RAS1:0#	
	0 1	RAS1#			
	1 0	RAS2#	10, 11	RAS3:2#	
	1 1	RAS3#			
11 (64 Mbyte DRAM per bank/leaf)	27:26		27:26		1000 0000H (128 Mbytes)
	0 0	RAS0#	00, 01	RAS1:0#	
	0 1	RAS1#			
	1 0	RAS2#	10, 11	RAS3:2#	
	1 1	RAS3#			

IMPLICATION: The MCU may assert RAS# to access a nonexistent 80960RP/RD DRAM bank. This may occur when the number of DRAM banks installed is less than the maximum number of DRAM banks used in the 80960RP/RD design. Two examples of when this problem can occur are:

- Re-mapping 80960RP/RD DRAM after DRAM accesses occurred in a previous memory map.
- Initializing from 80960RP/RD DRAM instead of using FLASH/ROM.

These two cases are described further in this erratum as CASE 1 - Re-mapping 80960RP/RD DRAM and CASE 2 - Initializing from 80960RP/RD DRAM instead of FLASH/ROM.

CASE 1 - Re-mapping 80960RP/RD DRAM

In an application using 1 Mbyte per bank/leaf and one Fast Page Mode (FPM) single-sided SIMM populated in a four bank design (total DRAM = 1 Mbyte), the following registers are set:

DRAM Bank Control Register (DBCR) = 0x0000 0001
DRAM Base Address Register (DBAR) = 0xD000 0000

The DBCR and DBAR values imply an address range of 1 Mbyte from 0xD000 0000 to 0xD00F FFFF.

The following sequence can occur:

1. A write is issued to 80960 local address 0xD000 1000.
2. Since 1-Mbyte is the DRAM bank/leaf size, the MCU decodes the next two higher order bits 21:20 from within the address to determine which RAS# signal to assert during the DRAM access.
3. Since address bits 21:20 = 00_2, the MCU asserts RAS0# (See Table 1).
4. The programmer then re-maps 80960RP/RD DRAM by programming the DBAR to 0xDFE0 0000. The address range is now 0xDFE0 0000 - 0xDFEF FFFF.
5. When a read is issued to 0xDFE0 1000 (i.e., the same address offset written in Step 1), the MCU asserts RAS2# because bits 21:20 = 10_2 (See Table 1). Data initially written to this location in Step 1 cannot be read.
6. Because the DRAM was remapped, the MCU now asserts RAS2# to an unpopulated DRAM bank and the data returned is invalid.

CASE 2 - Initializing from 80960RP/RD DRAM instead of FLASH/ROM

When the 80960RP/RD initializes from 80960 local memory instead of FLASH/ROM, the 80960RP/RD's first instruction fetch of the IBR is hard-coded to address 0xFEFF FF30. When the MCU reads this address, it asserts RAS2# or RAS3#, depending on the DRAM bank/leaf size.

In an application using 4 Mbyte per bank/leaf and two single-sided SIMMs populated in a four bank design (total DRAM = 8 Mbytes), the following registers are set:

DRAM Bank Control Register (DBCR) = 0x0000 0013
DRAM Base Address Register (DBAR) = 0xFE80 0000

The DBAR and DBCR values imply an address range of 8 Mbytes from 0xFE80 0000 to 0xFEFF FFFF.

The following sequence can occur:

1. A read of the IBR is issued to 0xFEFF FF30.
2. Since 4-Mbytes is the DRAM bank/leaf size, the MCU decodes the next two higher order bits 23:22 from within the address to determine which RAS# signal to assert during the DRAM access.
3. IBR address bits 23:22 = 11_2, and the MCU asserts RAS3# (See Table 1).
4. RAS3# selects an unpopulated DRAM bank, the IBR will not be read, and the device will not initialize.

WORKAROUND: Two workarounds are presented. The ***CASE 1 WORKAROUND*** describes a software modification. The ***CASE 2 WORKAROUND*** describes a hardware modification.

CASE 1 WORKAROUND - Re-mapping 80960RP/RD DRAM

When 80960RP/RD memory subsystem contains unpopulated DRAM banks, the DBAR must be aligned on an address boundary of a multiple of four times the DRAM bank/leaf size (for non-interleaved or interleaved memory) to ensure the correct RAS# is asserted (See Table 1). Limit 80960 local memory accesses to the total amount of memory installed in the system.

Note: This workaround is for booting from Flash/ROM. See ***CASE 2 WORKAROUND*** for booting from 80960RP/RD DRAM.

CASE 2 WORKAROUND - Initializing from 80960RP/RD DRAM instead of FLASH/ROM

In a standard DRAM configuration, RAS0# and RAS1# are routed to the front sides of the SIMMs, and RAS2# and RAS3# are routed to the back sides of the SIMMs. To implement this workaround, swap RAS0# and RAS2# and swap RAS1# and RAS3#. This routes RAS2# and RAS3# to the front sides of their respective SIMMs, and routes RAS0# and RAS1# to the back sides of their respective SIMMs (See Figure 1). To determine which RAS# is asserted for a particular address and DRAM configuration, see Table 1.

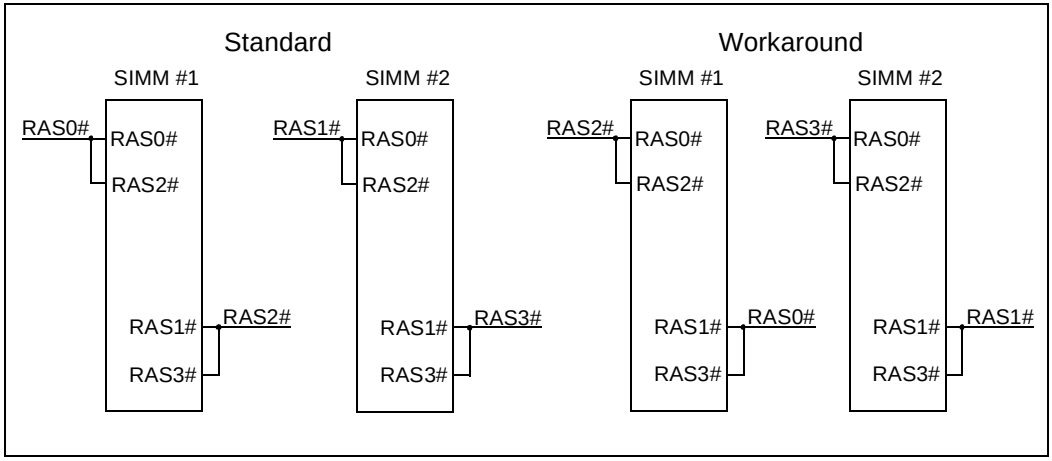


Figure 1. DRAM RAS# Configurations

In an application using 4 Mbytes per bank/leaf and two single-sided SIMMs populated in a four bank design (total DRAM = 8 Mbytes), the following registers are set:

DRAM Bank Control Register (DBCR) = 0x0000 0013
DRAM Base Address Register (DBAR) = 0xFE80 0000

The DBCR and DBAR values imply an address range of 8 Mbytes from 0xFE80 0000 to 0xFEFF FFFF.

When initializing from 80960RP/RD DRAM, the first instruction fetch of the IBR is hard-coded to address 0xFEFF FF30; as a result, A23:22 = 11_2 and RAS3# is asserted. With the workaround in place, the RAS# lines are swapped and RAS3# is connected to the front side of the second SIMM and the IBR can be read.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

24. ATU may deadlock when the PCI delayed read discard timer expires just as the delayed read completion cycle begins

PROBLEM: *PCI Local Bus Specification*, revision 2.1 requires that devices discard Delayed Requests when the initiating master doesn't repeat the request within 2^{15} (32768) clock cycles. This prevents deadlock conditions. The 80960RP/RD implements primary and secondary ATU discard timers. When the discard timer expires just as the initiating master repeats the PCI read request, the corresponding ATU can lock up. The device must be reset to recover from this condition.

This condition may occur for any delayed read transaction (i.e., MR, MRL, MRM); delayed write transactions are not affected. Both primary and secondary ATUs are affected.

IMPLICATION: When a PCI master has long delays to repeat a retried PCI read transaction, the ATU may deadlock when the read transaction is repeated just as the 80960RP/RD's discard timer expires.

WORKAROUND: The primary and secondary ATU discard timers can be programmed to expire in 2^{10} or 2^{15} clocks by setting bits 09:08 in the Bridge Control Register (BCR) at 80960 local bus address 103EH. The value selected should exceed the maximum time between retries of all masters that initiate PCI read transactions through either ATU.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

25. Changing the limit register when a value exists in the corresponding base address register may prevent access to the address space

PROBLEM: The 80960RP/RD provides a programmable mechanism for defining the memory block size requirements. This mechanism utilizes a base address register (BAR) and corresponding limit register. Any bit in a BAR becomes read-only when the corresponding bit in the associated limit register is cleared. When a bit is set in the BAR before the corresponding bit in the associated limit register is cleared, that bit in the BAR can no longer be cleared and remains set.

IMPLICATION: The address space defined by a BAR and limit register pair can become inaccessible if the limit register is changed to define a larger address space when the BAR has already been programmed to a non-zero value. This problem can exist with the following register pairs:

Register Name	Abbreviation	80960 local address
Primary Inbound Base Address Register	PIABAR	0x1210
Primary Inbound Limit Register	PIALR	0x1240
Secondary Inbound Base Address Register	SIABAR	0x1248
Secondary Inbound Limit Register	SIALR	0x124C
Expansion ROM Base Address Register	ERBAR	0x1230
Expansion ROM Limit Register	ERLR	0x1274

Since all bits in the BARs are used by the address detection logic, having a bit set (1) in the BAR, which is clear (0) in the corresponding limit register creates a condition where no PCI address is recognized as valid. For example:

Initial Settings:
PIALR = 0xFFFF F000 (default)
PIABAR = 0xFFA2 4000

When the PIALR is modified to 0xFFF**0 0**000 (bits 19:12 = 0), the PIABAR remains programmed to 0xFFA**2 4**000 (bits 19:12 are read only).

Inbound address detection is determined from the 32-bit PCI address, the base address register and the limit register. The algorithm for detection is:

When
PCI_Address & Limit_Register == Base_Register,
the PCI Address is claimed by the primary ATU.

WORKAROUND: Before programming the limit register to a larger block size, clear all bits of the corresponding BAR which are to be cleared (programmed to 0) in the limit register. For example:

Initial Settings:
PIALR = 0xFFFF F000 (default)
PIABAR = 0xFFA2 4000

To set the PIALR to 0xFFF**0 0**000 (bits 19:12 = 0), first program the PIABAR to 0xFFA**0 0**000 (or some larger address boundary — at least bits 19:12 = 0).

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

26. Messaging Unit may drop messages during simultaneous inbound primary PCI bus reads and 80960 local bus reads

PROBLEM: The MU's Outbound Post Queue (OPQ) holds outbound messages from the i960 core processor for the other processors to process. The queue utilizes an Outbound Posted Head Pointer (OPHP) and Outbound Posted Tail Pointer (OPTP) to keep track of message pointers in the queue.

- When the queue is empty, the OPHP and OPTP are equal and the value of FFFF FFFFH is returned on the primary PCI bus.
- When the queue is not empty, the OPHP and OPTP are not equal, data is returned on the primary PCI bus and the OPTP is incremented by MU hardware.

A boundary condition exists when the OPQ is transitioning from the empty state (OPHP == OPTP) to a non-empty state and a PCI memory read to the OPQ occurs during the same clock. Under this condition, the MU can drop a message that was bound for the primary PCI bus.

This condition also affects the Inbound Free Queue (IFQ).

IMPLICATION: The Messaging Unit can drop Outbound Post Queue and Inbound Free Queue messages destined for the primary PCI bus.

WORKAROUND: To prevent the MU from dropping messages, the 80960RP/RD programmer must add the following code to the routine where the messages are being loaded to 80960 local memory.

The following workaround is valid for the Outbound Post Queue and the Inbound Free Queue:

1. Temporarily disable the primary ATU by disallowing the primary ATU from obtaining GNT# from the 80960RP/RD local bus arbiter.
   a. Save the state of the Local Bus Arbitration Control Register (LBACR - 1600H).
   b. Set bits 11:10 = 11_2 in the LBACR.
2. Check for the OPQ [IFQ] empty state, OPHP == OPTP [Inbound Free Head Pointer == Inbound Free Tail Pointer].
3. When the OPQ [IFQ] is empty,
   a. Write a null message (FFFF FFFFH) to the end of the OPQ [IFQ].
   b. Write a valid message to the next location of the OPQ [IFQ].
   c. Increment the OPHP [IFHP] by eight using a `long int`.
4. When the OPQ [IFQ] is not empty,
   a. Write a valid message to the end of the OPQ [IFQ].

b. Increment the OPHP [IFHP] by four.

5. Allow the primary ATU to obtain GNT# from the 80960RP/RD local bus arbiter by restoring the LBACR to its original value saved in Step 1a.

After a primary PCI bus interrupt is generated and to ensure the OPQ is empty, it is beneficial to reduce interrupt latency by reading the Outbound Queue Port from the primary PCI bus a second time when a value of FFFF FFFFH is read.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

27. P_REQ# is not deasserted when a single DWORD transfer is retried

PROBLEM: When the 80960RP/RD is mastering a single DWORD transaction on the primary PCI bus and it is retried, P_REQ# will not deassert after the retry. P_REQ# remains asserted until the transaction completes or aborts.

IMPLICATION: When the host system has not implemented arbitration that conforms to a fairness algorithm on the 80960RP/RD's primary PCI bus, the 80960RP/RD will continue to own the bus and enter into a deadlock condition.

WORKAROUND: *PCI Local Bus Specification*, revision 2.1 section 3.4 states that arbiters are required to implement a fairness algorithm. Make certain that the 80960RP/RD design is used in a host system compliant to the Arbitration section of the Specification.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

28. Bit 0 of the ATURID register is permanently set to a one

PROBLEM: Bit 0 of the ATU Revision ID register is permanently set to a one.

IMPLICATION: The ATURID register is a read/write register from the 80960 local bus. Since bit 0 of the ATURID is always set, the bit operates as a read only bit. Writing any value to bit 0 will always read back a one. Bits 7:1 of the ATURID remain read/write from the 80960 local bus.

WORKAROUND: There is no workaround.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

29. VGA memory addresses are aliased throughout the PCI memory address range by the PCI-to-PCI Bridge when the VGA Enable bit is set.

PROBLEM: When the VGA Enable bit is set (Bit 03 in the Bridge Control Register, local bus address 103EH) the PCI-to-PCI bridge should positively decode any PCI memory address that falls in the range of 0x000A 0000H - 0x000 BFFFFH and forward them from the primary to secondary interface. However, the problem exists that the PCI-to-PCI bridge will positively decode and forward ANY memory address that falls within the range of 0xXX0A 0000H - 0xXX0B FFFFH. These addresses are also blocked from being claimed by the secondary PCI interface and will not be forwarded upstream to the primary interface. Because the upper 8 bits of the PCI memory address is not decoded by the PCI-to-PCI bridge, the VGA region is aliased 256 times within the 4 Gigabyte PCI memory address space.

For example, the bridge will incorrectly decode 0xF00A 0000H as a VGA memory address. This decode is completely independent of how the bridge prefetchable and nonprefetchable memory decode registers are configured (Memory limit @ 20H, Memory Base @ 24H, Prefetchable memory limit @ 24H, Prefetchable memory base @ 28H).

IMPLICATION: If the VGA Enable bit is set in the Bridge Control Register, the PCI-to-PCI bridge will improperly forward memory addresses from the primary to secondary PCI bus interface that fall within the aliased memory regions. Conversely, the PCI-to-PCI bridge will also NOT correctly forward memory addresses from the secondary PCI bus to the primary PCI bus interfaces if the addresses fall within the aliased memory regions.

WORKAROUND: There is no workaround.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

30. PCI memory addresses that are aliases of VGA I/O addresses will not be forwarded from the secondary to the primary PCI Bus interface if the VGA Enable bit is set.

PROBLEM: When the VGA Enable bit is set (Bit 03 in the Bridge Control Register, local bus address 103EH) the PCI-to-PCI bridge's secondary PCI bus interface will not claim any PCI memory address that falls in the range of 0xXXXX X3B0 - 0xXXXX X3BBH or 0xXXXX X3C0 - 0xXXXX X3DFH and forward the cycle to the primary interface.

This erratum only effects the secondary PCI bus interface, these addresses ARE NOT forwarded from the primary to secondary interface. *This erratum is completely independent of how the bridge I/O Base Registers and I/O Limit Registers are programmed.

IMPLICATION: If the VGA Enable bit is set in the Bridge Control Register, then the PCI-to-PCI bridge will not forward memory addresses that fall within the aliased memory regions from the secondary PCI bus interface to the primary PCI bus.

WORKAROUND: There is no workaround.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

31. Inbound ATU writes to non-existent 80960 local memory will cause the next PCI configuration write cycle to target abort on the PCI bus.

PROBLEM: The 80960RP/RD's memory controller has a bus monitor feature which asserts LRDYRCV# if valid data is not returned for 80960 local bus accesses in 127 CLKIN periods. The bus monitor feature keeps the local bus from deadlocking if a local bus cycle addresses an invalid memory address (one that doesn't return LRDYRCV# or nonexistent memory space). If the bus monitor expires for an inbound write cycle through either the secondary ATU or the primary ATU, the next PCI configuration write cycle through that same ATU will target abort on the PCI bus. Note that even though the PCI configuration cycle target aborted, the appropriate address in configuration space is still written correctly.

IMPLICATION: If inbound ATU writes address local bus addresses that do not return LRDYRCV#, the next inbound PCI configuration write cycle will cause a target abort on the PCI bus. The implications of target aborts are system dependent.

WORKAROUND: Ensure that inbound ATU write cycles always address local bus memory space that will return LRDYRCV#. This can be done by programming the ATU Inbound Limit Registers (PIALR, SIALR) and the Inbound Translate Value Registers (PIATVR,SIATVR) to define a window to a region in 80960 local memory space that always returns LRDYRCV#, this will prevent the bus monitor timer from expiring.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

32. Inbound configuration write cycles may latch invalid data on the PCI bus if STOP# is asserted before the initiator asserts IRDY# during the delayed request cycle.

PROBLEM: All inbound configuration write cycles are treated as delayed transactions. During the delayed request cycle, the 80960RP/RD (PCI target) latches valid data on the PCI bus and retries the initiator by asserting STOP#. According to the Target Termination Signaling rules in the PCI Local Bus Specification Revision 2.1 (Section 3.3.3.2.1), once an initiator sees STOP# asserted, the initiator first must assert IRDY# and deassert FRAME# on the first cycle after IRDY# is asserted. It is recommended that IRDY# be asserted as soon as possible after STOP#. In the case of the target asserting STOP# before the initiator asserts IRDY#, the initiator is not required (although recommended) to provide valid data on the PCI bus when IRDY# is asserted.

The problem is that the 80960RP/RD does not recognize this particular case for configuration writes and treats it as the delayed request cycle and latches data on the PCI bus. See the timing diagram below. If the PCI master begins the cycle by inserting waitstates (IRDY# asserted) before STOP# is asserted and doesn't drive valid data on the PCI bus when IRDY# is asserted, the ATU will incorrectly latch invalid data and write to the PCI configuration register.

Only inbound PCI configuration cycles are affected by this errata. The ATU does not treat PCI memory writes and Memory write-invalidate as delayed transactions.

IMPLICATION: When used with PCI initiators that can assert IRDY# with invalid data for PCI configuration writes under the conditions described above, the 80960RP/RD can write invalid data to a PCI configuration register.

Note that for the ATU to retire the delayed completion cycle, the initiator must reissue the original request with the same data. If the initiator reissues the initial request and asserts IRDY# before STOP# with valid data this time, the data will not match and the delayed completion cycle will not be retired. Potentially, the reissued cycle can be retried until the discard timer expires. Once the discard timer expires, the cycle is accepted (this time with valid data) and the correct data gets written into the PCI configuration register.

WORKAROUND: Do not use the 80960RP/RD with PCI initiators that insert IRDY# waitstates during PCI configuration cycles and drive invalid data on the bus when IRDY# is asserted following STOP#.

STATUS: For the steppings affected see the SUMMARY TABLE OF CHANGES.

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.

For the 80960RP 33/5.0, 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* 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 (ball AD8) 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 09:05 of the Secondary IDSEL Select Register (SISR) can be used to mask the S_AD25:21 during Type 0 and Type 1 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) 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.

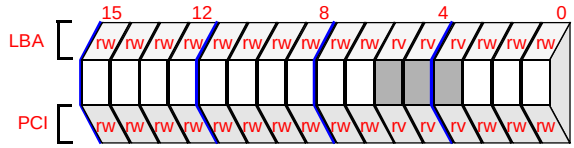
- When cleared (0), deassertion period is one-half of a CLKIN period.
- When set (1), the deassertion period is extended by the wait state profile defined in the MBWWSx registers in addition to the one-half clock in period. Also when set, the MA11:0 and BE1:0 keep their current state for one clock after 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) 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 (Sheet 1 of 2)

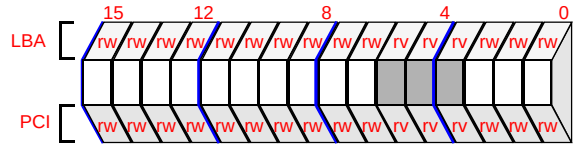
PCI Configuration Address Offset: 5CH
80960 Core Local Bus Address: 105CH



Bit	Default	Description
15[1]	0_2	S_SERR# Detected Interrupt Mask - When set, detecting S_SERR# on the secondary interface resulting in bit 14 of the SSR being set will *not* result in bit 4 of the SBISR being set. When clear, an error that sets bit 14 of the SSR will cause bit 4 of the SBISR to be set
14[1]	0_2	Secondary PCI Master Abort Interrupt Mask - When set, a master abort error resulting in bit 13 of the SSR being set will *not* result in bit 3 of the SBISR being set. When clear, an error that sets bit 13 of the SSR will cause bit 3 of the SBISR to be set.
13[1]	0_2	Secondary PCI Target Abort (Master) Interrupt Mask- When set, a target abort error resulting in bit 12 of the SSR being set will *not* result in bit 2 of the SBISR being set. When clear, an error that sets bit 12 of the SSR will cause bit 2 of the SBISR to be set.
12[1]	0_2	Secondary PCI Target Abort (Target) Interrupt Mask - When set, a target abort error resulting in bit 11 of the SSR being set will *not* result in bit 1 of the SBISR being set. When clear, an error that sets bit 11 of the SSR will cause bit 1 of the SBISR to be set.
11[1]	0_2	Secondary PCI Master Parity Error Interrupt Mask - When set a parity error resulting in bit 8 of the SSR being set will *not* result in bit 0 of the SBISR being set. When clear, an error that sets bit 8 of the SSR will cause bit 0 of the SBISR to be set.
10[1]	0_2	P_SERR# Asserted Interrupt Mask - When set, detecting or asserting P_SERR# on the primary interface resulting in bit 14 of the PSR being set will *not* result in bit 4 of the PBISR being set. When clear, an error that sets bit 14 of the PSR will cause bit 4 of the PBISR to be set.
09[1]	0_2	Primary PCI Master Abort Interrupt Mask - When set, a master abort error resulting in bit 13 ofthe PSR being set will *not* result in bit 3 of the PBISR being set. When clear, an error that sets bit 13 of the PSR will cause bit 3 of the PBISR to be set.

Table 10-1. Secondary Decode Enable Register - SDER (Sheet 2 of 2)

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



Bit	Default	Description
08[1]	0_2	Primary PCI Target Abort (Master) Interrupt Mask- When set, a target abort error resulting in bit 12 of the PSR being set will *not* result in bit 2 of the PBISR being set. When clear, an error that sets bit 12 of the PSR will cause bit 2 of the PBISR to be set.
07[1]	0_2	Primary PCI Target Abort (Target) Interrupt Mask - When set, a target abort error resulting in bit 11 of the PSR being set will *not* result in bit 1 of the PBISR being set. When clear, an error that sets bit 11 of the PSR will cause bit 1 of the PBISR to be set.
06[1]	0_2	Primary PCI Master Parity Error Interrupt Mask - When set a parity error resulting in bit 8 of thePSR being set will *not* result in bit 0 of the PBISR being set. When clear, an error that sets bit 8 of the PSR will cause bit 0 of the PBISR to be set.
05[1]	0_2	Reserved
04[1]	0_2	Reserved
03	0_2	Reserved
02	0_2	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	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	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.

1. Bits 15:04 do not exist for the 5.0 Volt device.

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

ITEM: The 80960RP/RD provides a programmable mechanism for defining the memory address block size requirements. This mechanism uses the Base Address Register (BAR) and corresponding limit register. 80960RP/RD 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 FFFFH or FFFF FFFEH).

12. Burst EDO (BEDO) memories are no longer supported

ITEM: Burst EDO memories are no longer supported. The DRAM Bank Type/Arrangement field (bits 7:3) in the DRAM bank control Register (local bus address 151CH) must must never be programmed with the following values: 1xx00, 1xx01, 1xx1x.

13. The I/O APIC is no longer supported

ITEM: The functionality of the I/O APIC is no longer supported. These specific registers are no longer supported and should be treated as reserved memory space. The primary PCI interrupt pins should be used as the mechanism for routing interrupts. The following registers are associated with the I/O APIC and must not be used:

- APIC ID Register (local bus address 1780H)
- APIC Arbitration Register (local bus address 1784H)
- EOI Vector Register (local bus address 1788H)
- Interrupt Message Register (local bus address 178CH)
- APIC Control/Status Register (local bus address 1790H)
- APIC Register Select Register (offset 0 in the Primary ATU Address Space)
- APIC Window Register (offset 8H in the Primary ATU Address Space)

To ensure proper operation PICCLK, PICD1, PICD0 should be pulled up to Vcc or down to Vss.

14. The PCI-to-PCI bridge no longer supports secondary positive memory decoding

ITEM: The PCI-to-PCI bridge no longer supports secondary positive memory decoding. The Secondary Positive Memory Decode Enable bit (bit 1, in the Secondary Decode Enable Register, local bus address 105CH) is reserved and must never be set to 1.

15. The PCI-to-PCI bridge no longer supports secondary positive I/O decoding

ITEM: The PCI-to-PCI bridge no longer supports secondary positive I/O decoding. The Secondary Positive I/O Decode Enable bit (bit 0, in the Secondary Decode Enable Register, local bus address 105CH) is reserved and must never be set to 1.

16. Secondary PCI Boot Mode

ITEM: The ATU no longer supports booting from the secondary PCI bus. The Secondary PCI Boot Mode bit (bit 11, in the ATU Configuration Register, local bus address 1288H) must never be set to 1.

17. 32-bit PCI expansion ROM support

ITEM: The Primary ATU no longer supports 32-bit expansion ROMs. Only 8-bit expansion ROMs are supported. The Expansion ROM Width (bit 06, in the ATU configuration Register, local bus address 1288H) should always be cleared to 0.

18. 64-bit Dual Address Cycles (DACs) are not supported

ITEM: The PCI-to-PCI bridge, DMA controller, and Address Translation Units no longer support Dual Address Cycles

PCI-to-PCI bridge - The DAC Cycle Enable bit (Bit 06, in the Extended Bridge Control Register, local bus address 1040H) must always be set to 1. In addition, PCI devices installed on the secondary PCI bus should never generate 64-bit DAC cycles.

DMA controller - The DMA controllers must never be programmed to generate 64-bit DAC cycles. The PCI Upper Address Register in the DMA descriptor must always be programmed to 0x0000h.

Address Translation Units - The Primary DAC and Secondary DAC Outbound Address Translation Windows are not supported. The following registers are reserved:

- Primary Outbound DAC Window Value Reg (local bus address 1260H)
- Primary Outbound Upper 64-bit DAC Reg (local bus address 1264H)
- Secondary Outbound DAC Window Value Reg (local bus address 1260H)

- Secondary Outbound Upper 64-bit DAC Reg (local bus address 1264H)

19. The forwarding of configuration writes from the secondary PCI bus through the PCI-to-PCI bridge is not supported

ITEM: The bridge will not support the forwarding of type1 configuration writes from the secondary PCI bus to the primary PCI bus. These configuration writes are only used for the forwarding of PCI special cycles from the secondary PCI bus to the primary PCI bus.

20. Messaging Unit Index Registers are no longer supported

ITEM: The Index Registers in the Messaging Unit are no longer supported. The Index Register Interrupt Mask bit (bit 06, in the Inbound Interrupt Mask Register, local bus address 1328H) should always be set to 1 to mask the index register interrupt if your applications writing to the index register space in the first 4K of the Primary ATU Address Space.

SPECIFICATION CLARIFICATIONS

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

ITEM: Bits 09:05 in the Secondary IDSEL Select Register (SISR) may be used in future versions to enable the masking of S_AD25:21 during Type 0 and Type 1 configuration cycles on the secondary PCI bus. These five signals (in addition to S_AD20:16) will enable devices as public or private PCI devices in an intelligent I/O design.

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

ITEM: When using an ICE in your 80960RP/RD 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.

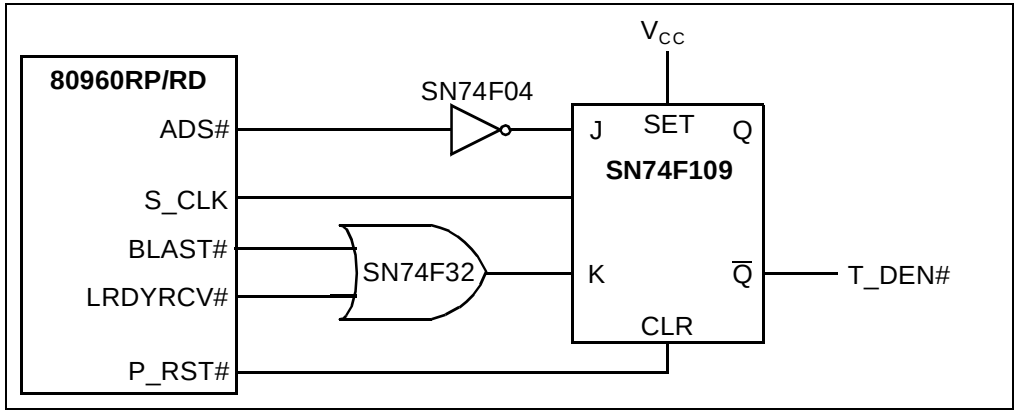
Using DEN# *with an In-Circuit Emulator:* For ICE users, it is not recommended that you use the 80960RP/RD's DEN# signal directly to your transceivers**.** When executing an ICE microcode transaction, the expected behavior is that DEN# would remain de-asserted during the entire transaction. However, 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 80960RP/RD 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 80960RP/RD'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 80960RP/RD 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 80960RP/RD 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 80960RP/RD 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 80960RP/RD'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 80960RP/RD'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 80960RP/RD asserts PERR# and does not propagate the write through to the target bus. During the delayed write completion cycle, the 80960RP/RD 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 80960RP/RD 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 80960RP/RD provides a programmable mechanism for defining the memory block size requirements. This mechanism uses the Base Address Register (BAR) and corresponding limit register. 80960RP/RD 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 80960RP/RD'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 80960RP/RD'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 80960RP/RD registers, queues, and units. Clearing the bit results in the S_RST# output to deassert.

The 80960RP/RD 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 80960RP/RD registers, two possible workarounds are:

1. Connect the 80960RP/RD's S_RST# output to the 80960RP/RD'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 80960RP/RD 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 80960RP/RD. 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 80960RP/RD'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).

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).

9. Determining Block Sizes for Base Address Registers

ITEM: This clarification is an update to, and summary of, Specification Changes 4, 6, and 11, and Specification Clarifications 5 and 6.

The Primary and Secondary Inbound ATU Base Address Registers and Expansion ROM Base Address Register use their associated limit registers for defining the requested address space size. The requested address space size and type can be determined by writing all 1’s to a base address register, and then reading back from the register as described in Section 6.2.5.1 of the *PCI Local Bus Specification*, revision 2.1. Table A describes the device specific values used to determine the memory block size. By scanning the bits of the returned value of the base address register in ascending order starting with bit 04, the programmer can determine the required address space size. The binary-weighted value of the first set (1) bit found indicates the required amount of space. Table 9-1 describes the relationship between the values read back and the byte sizes the base address register requires.

Table 9-1. Device Specific Instructions for Base Address Register (Sheet 1 of 2)

Device Part Number	Value Written to the BAR	Effect of writing the value to the Base Address Register
80960RP 33/5.0 A-0	FFFF FFFFH	The first read after a write of FFFF FFFFH to the base address register is directed to the limit register. The data returned on subsequent reads from the base address register is the contents of the base address register — not the contents of the corresponding limit register. If any other value is written to the base address register, that value is programmed into the base address register. When determining block size requirements, reading to or writing from the base address register must be using 32-bit configuration cycles. However, configuration cycles not used to determine block size requirements can be performed as 8-, 16-, or 32-bit accesses.
80960RP 33/5.0 A-1	FFFF FFFEH or FFFF FFFFH	The first read after a write of FFFF FFFEH or FFFF FFFFH to the base address register is directed to the limit register. The data returned on subsequent reads from the register is the contents of the base address register — not the contents of the corresponding limit register. If any other value is written to the base address register, that value is programmed into the base address register. When determining block size requirements, reading to or writing from the base address register must be using 32-bit configuration cycles. However, configuration cycles not used to determine block size requirements can be performed as 8-, 16-, or 32-bit accesses.
80960RP 33/3.3 A-0 80960RD 66/3.3 A-0	FFFF FFFEH or FFFF FFFFH	The read after a write of FFFF FFFEH or FFFF FFFFH to the base address register is directed to the limit register. The data returned on subsequent reads from the base address register returns the limit register contents until the base address register is rewritten with a value other than FFFF FFFEH or FFFF FFFFH. If any other value is written to the base address register, that value is programmed into the base address register. When determining block size requirements, reading to or writing from the base address register must be using 32-bit configuration cycles. However, configuration cycles not used to determine block size requirements can be performed as 8-, 16-, or 32-bit accesses.

Table 9-1. Device Specific Instructions for Base Address Register (Sheet 2 of 2)

Device Part Number	Value Written to the BAR	Effect of writing the value to the Base Address Register
80960RP 33/3.3 B-0 80960RD 66/3.3 B-0	FFFF FFFEH or FFFF FFFFH	The limit register is a bitwise enable of the base address register from bits 31:12. When limit register bits (bits 31:12) are set to a 1, the corresponding bits in the base register are enabled as read/write. When limit register bits (bits 31:12) are cleared to a 0, the corresponding bits in the base register are cleared to zero and are read-only. Once the limit register is programmed, FFFF FFFEH or FFFF FFFFH can be written to the base address register, and the memory block size requirement can be found by reading back the base address register. Values used for programming the limit register should be similar to those listed in Table 9-2. Any access, including configuration cycles, to the base address register can be performed as an 8-, 16-, or 32-bit access to determine block size requirements.

Table 9-2. Memory Block Size Read Response (Sheet 1 of 2)

Response After Writing all 1's to the Base Address Register	Block Size
FFFF F00xH	4 Kbytes
FFFF E00xH	8 Kbytes
FFFF C00xH	16 Kbytes
FFFF 800xH	32 Kbytes
FFFF 000xH	64 Kbytes
FFFE 000xH	128 Kbytes
FFFC 000xH	256 Kbytes
FFF8 000xH	512 Kbytes
FFF0 000xH	1 Mbytes
FFE0 000xH	2 Mbytes
FFC0 000xH	4 Mbytes
FF80 000xH	8 Mbytes
FF00 000xH	16 Mbytes

Table 9-2. Memory Block Size Read Response (Sheet 2 of 2)

Response After Writing all 1's to the Base Address Register	Block Size
FE00 000xH	32 Mbytes
FC00 000xH	64 Mbytes
F800 000xH	128 Mbytes
F000 000xH	256 Mbytes
E000 000xH	512 Mbytes
C000 000xH	1 Gbytes
8000 000xH	2 Gbytes
0000 0000H	Register not implemented, no address space required.

As an example, assume that FFFF FFFFH is written to the ATU Primary Inbound Base Address Register (PIABAR) and the value read back is FFF0 0008H. Bit 0 is a zero, so the device requires memory address space. Bits 2:1 are 00_2, so the memory can be located anywhere within 32-bit address space (4 Gbytes). Bit 3 is a one, so the memory does support prefetching. Scanning upwards starting at bit 4, bit 20 is the first bit set to one that is encountered. The binary-weighted value of this bit is 1,048,576 which indicates that the device requires 1 Mbyte of memory space.

DOCUMENTATION CHANGES

1. Section 8.3.2, Page 8-26

The text in this section is incomplete and the XINT Select Bit values are interchanged in Table 8-2 and Table 8-3. Replace the entire section with the following text and tables:

Four PCI interrupt inputs can be routed to either the i960 core processor interrupt inputs or to the primary PCI bus interrupt output pins. This routing is controlled by the XINT Select Bits in the PCI Interrupt Routing Select Register (See Table 8-8 and Table 8-9). Table 8-2 and Table 8-3 below are summaries of the PIRSR register definitions.

Table 8-2. PCI Interrupt Routing Summary for 80960RP 33/5.0 Volt

XINT Select Bit	Bit Value	Description
00	1	**S_INTD#/XINT3#** interrupt pin is routed to the i960 core processor interrupt controller **XINT3#** input **S_INTC#/XINT2#** interrupt pin is routed to the i960 core processor interrupt controller **XINT2#** input **S_INTB#/XINT1#** interrupt pin is routed to the i960 core processor interrupt controller **XINT1#** input **S_INTA#/XINT0#** interrupt pin is routed to the i960 core processor interrupt controller **XINT0#** input
	0	**S_INTD#/XINT3#** interrupt pin is routed to the **P_INTD#** pin **S_INTC#/XINT2#** interrupt pin is routed to the **P_INTC#** pin **S_INTB#/XINT1#** interrupt pin is routed to the **P_INTB#** pin **S_INTA#/XINT0#** interrupt pin is routed to the **P_INTA#** pin

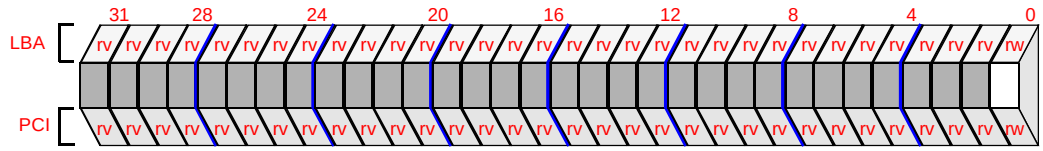
Table 8-3. PCI Interrupt Routing Summary for 80960RP/RD 33/66/3.3 Volt

XINT Select Bit	Bit Value	Description
03	1	**S_INTD#/XINT3#** interrupt pin is routed to the i960 core processor interrupt controller **XINT3#** input
	0	**S_INTD#/XINT3#** interrupt pin is routed to the **P_INTD#** output pin
02	1	**S_INTC#/XINT2#** interrupt pin is routed to the i960 core processor interrupt controller **XINT2#** input
	0	**S_INTC#/XINT2#** interrupt pin is routed to the **P_INTC#** output pin
01	1	**S_INTB#/XINT1#** interrupt pin is routed to the i960 core processor interrupt controller **XINT1#** input
	0	**S_INTB#/XINT1#** interrupt pin is routed to the **P_INTB#** output pin
00	1	**S_INTA#/XINT0#** interrupt pin is routed to the i960 core processor interrupt controller **XINT0#** input
	0	**S_INTA#/XINT0#** interrupt pin is routed to the **P_INTA#** output pin

2. Section 8.4.1, Page 8-32, Table 8-8

The values in the definition of bit 00 are interchanged. The correct register definition is:

Table 8-8. PCI Interrupt Routing Select Register – PIRSR (80960RP 33/5.0 Volt)



LBA: 1050H
PCI: 50H

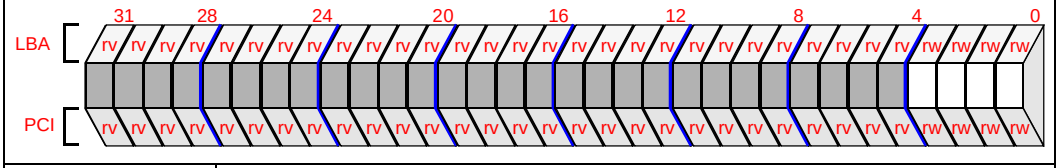
Legend:
NA = Not Accessible
RO = Read Only
RV = Reserved
PR = Preserved
RW = Read/Write
RS = Read/Set
RC = Read Clear
LBA = 80960 local bus address
PCI = PCI Configuration Address Offset

Bit	Default	Description
31:01	0000 0000H	Reserved. Initialize to 0.
00	0_2	XINT Select Bit - (1) S_INTD#/XINT3# interrupt pin is routed to the i960 core processor interrupt controller XINT3# input S_INTC#/XINT2# interrupt pin is routed to the i960 core processor interrupt controller XINT2# input S_INTB#/XINT1# interrupt pin is routed to the i960 core processor interrupt controller XINT1# input S_INTA#/XINT0# interrupt pin is routed to the i960 core processor interrupt controller XINT0# input (0) S_INTD#/XINT3# interrupt pin is routed to the P_INTD# pin S_INTC#/XINT2# interrupt pin is routed to the P_INTC# pin S_INTB#/XINT1# interrupt pin is routed to the P_INTB# pin S_INTA#/XINT0# interrupt pin is routed to the P_INTA# pin

3. Section 8.4.1, Page 8-32, Table 8-9

The values in the definition of bits 03:00 are interchanged. The correct register definition follows:

Table 8-9. PCI Interrupt Routing Select Register – PIRSR (80960RP/RD 33/66/3.3 Volt)



LBA: 1050H
PCI: 50H

Legend:
RV = Reserved
RS = Read/Set
LBA = 80960 local bus address
NA = Not Accessible
PR = Preserved
RC = Read Clear
PCI = PCI Configuration Address Offset
RO = Read Only
RW = Read/Write

Bit	Default	Description
31:04	0000 000H	Reserved. Initialize to 0.
03	0_2	XINT3 Select Bit - (1) S_INTD#/XINT3# interrupt pin is routed to the i960 core processor interrupt controller XINT3# input (0) S_INTD#/XINT3# interrupt pin is routed to the P_INTD# output pin
02	0_2	XINT2 Select Bit - (1) S_INTC#/XINT2# interrupt pin is routed to the i960 core processor interrupt controller XINT2# input (0) S_INTC#/XINT2# interrupt pin is routed to the P_INTC# output pin
01	0_2	XINT1 Select Bit - (1) S_INTB#/XINT1# interrupt pin is routed to the i960 core processor interrupt controller XINT1# input (0) S_INTB#/XINT1# interrupt pin is routed to the P_INTB# output pin
00	0_2	XINT0 Select Bit - (1) S_INTA#/XINT0# interrupt pin is routed to the i960 core processor interrupt controller XINT0# input (0) S_INTA#/XINT0# interrupt pin is routed to the P_INTA# output pin

4. Section 11.7.4, Page 11-27

Replace the 0.01 µF value with 0.1 µF in the last sentence of the second paragraph. It should now read:

The 0.1 µF capacitor must be of the type X7R and the node connecting V_{CCPLL} must be as short as possible.

5. Section 11.7.4, Page 11-27, Figure 11-6

Replace Figure 11-6 with the following figure:

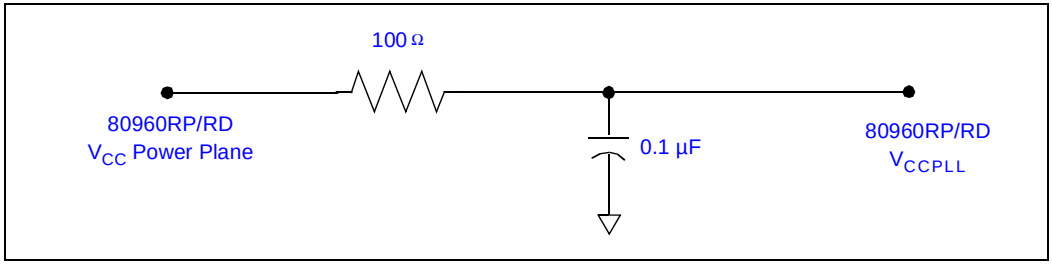
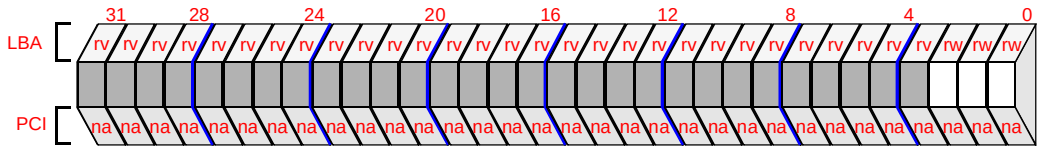


Figure 11-6. V_{CCPLL} Lowpass Filter

6. Section 12.3.1, Page 12-6, Table 12-3

Bit 00 and bit 02 descriptions are incorrectly interchanged. The correct register description for the Bus Control Register (BCON) is shown below:

Table 12-3. Bus Control Register – BCON



LBA: 86FCH **PCI:** NA	**Legend:** RV = Reserved RS = Read/Set LBA = 80960 Local Bus Address	NA = Not Accessible PR = Preserved RC = Read Clear PCI = PCI Configuration Address Offset

Bit	Default	Description
31:03	0000 0000H	Reserved.
02	0_2	Supervisor Internal RAM Protection (0) = First 64 bytes not protected from supervisor mode write (1) = First 64 bytes protected from supervisor mode writes
01	0_2	Internal RAM Protection (0) = Internal data RAM not protected from user mode writes (1) = Internal data RAM protected from user mode write
00	0_2	Configuration Entries in Control Table Valid) (0) = PMCON entries not valid, default to PMCON14_15 setting (1) = PMCON entries valid

7. Section 14.4, Page 14-5, Figure 14-3

Figure 14-3 is incorrect. The correct figure is shown below:

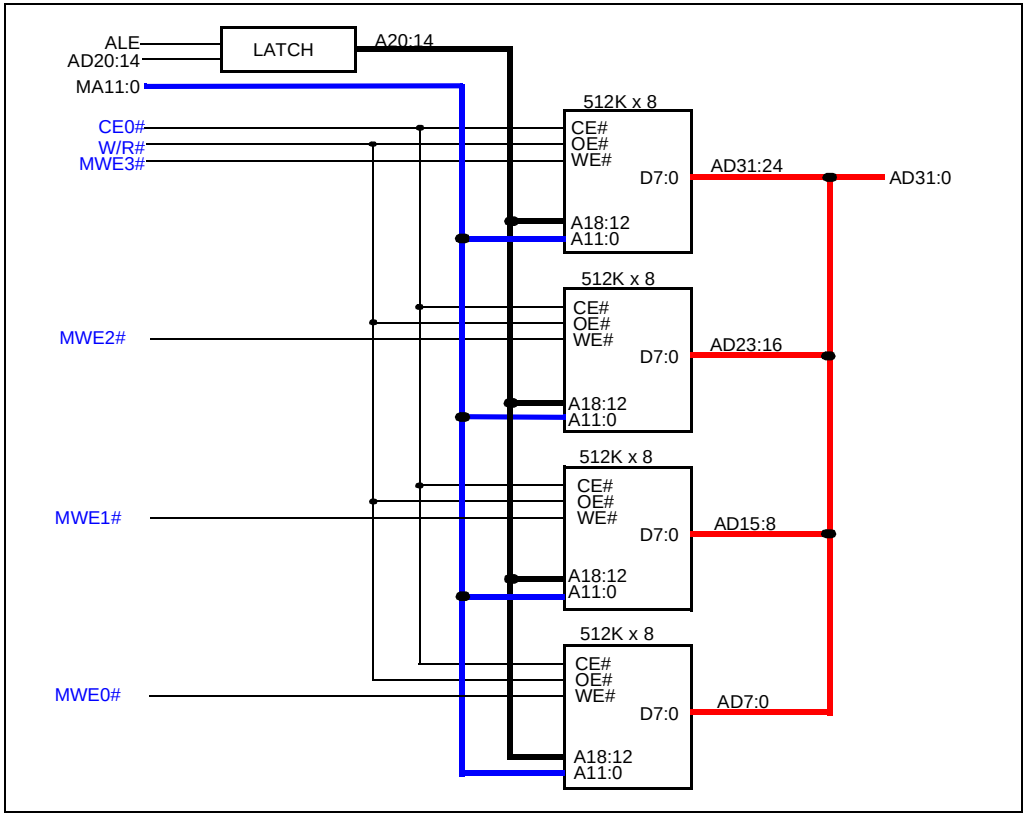


Figure 14-3. 32-Bit ROM or SRAM System

8. Section 14.6.7, Page 14-31, Table 14-20

The DRAM Bank Write Wait State Register (DWWS) bits 17:16 values are incorrect. It currently reads:

Bit	Default	Description
17:16	00_2	**DRAM Write cycle RAS-to-CAS delay (t_{WRC})** - This field affects the number of cycles between the assertion of RAS3:0# and the assertion of CAS7:0#.

	FPM	EDO	BEDO
00	1.5 cycles	1 cycles	1 cycles
01	2.5 cycles	2 cycles	2 cycles
10	3.5 cycles	3 cycles	3 cycles
11	4.5 cycles	4 cycles	4 cycles

It should read:

Bit	Default	Description
17:16	00_2	**DRAM Write cycle RAS-to-CAS delay (t_{WRC})** - This field affects the number of cycles between the assertion of RAS3:0# and the assertion of CAS7:0#.

	FPM	EDO	BEDO
00	1.5 cycles	1.5 cycles	1.5 cycles
01	2.5 cycles	2.5 cycles	2.5 cycles
10	3.5 cycles	3.5 cycles	3.5 cycles
11	4.5 cycles	4.5 cycles	4.5 cycles

9. Section 16.7.12, Page 16-39

In the first sentence, FFF0 0004H should correctly read FFF0 0008H.

In the third sentence, the 2 in 002 should be subscripted. The sentence should now read:

Bits 2:1 are 00_2, so the memory can be located anywhere within 32-bit address space (4 Gbytes).

10. Section 16.7.21, Page 16-47, Table 16-31

In Table 16-31 the default value for bit 31:02 is incorrectly stated to be 0001 000H. The default value for bit 31:02 is 0000 100H 00_2. The default value for bit 12 is one.

11. Section 15.13.34, Page 15-75, Table 15-45

The footnote of Table 15-45 is incorrect. It currently reads:

Bits 15:06 do not exist for the 5.0 Volt device.

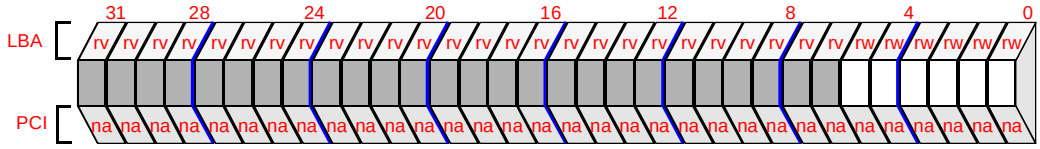
It should read:

Bits 15:04 do not exist for the 5.0 Volt device.

12. Section 17.7.11, Page 17-26, Table 17-15

The last two rows in Table 17-15 should be transposed.

Table 17-15 Messaging Unit Configuration Register - MUCR



LBA: 1350H PCI: NA	**Legend:** RV = Reserved RS = Read/Set LBA = 80960 Local Bus Address Offset	NA = Not Accessible PR = Preserved RC = Read Clear PCI = PCI Configuration Address

Bit	Default	Description
31:05	0000 00H	Reserved.
05:01	1H	Circular Queue Size - This field determines the size of each Circular Queue. All four queues are the same size. Circular Queue Size: • (00001_2) 4K entries (16 Kbytes) • (00010_2) 8K entries (32 Kbytes) • (00100_2) 16K entries (64 Kbytes) • (01000_2) 32K entries (128 Kbytes) • (10000_2) 64K entries (256 Kbytes)
00	0_2	Circular Queue Enable - This bit enables or disables the Circular Queues. When clear, Circular Queues are disabled. The MU accepts PCI accesses to the Circular Queue Ports; the MU ignores data for Writes and returns FFFF FFFFH for Reads. When set, Circular Queues are enabled.

13. Section 16.7.18, Page 16-44, Table 16-28

In the ATU Minimum Grant Register the values in the bits 07:00 should be Read Only bits, the table currently shows those bits to be Read/Write bits.

14. Section 16.7.19, Page 16-45, Table 16-29

In the ATU Maximum Latency Register the values in the bits 07:00 should be Read Only bits, the table currently shows those bits to be Read/Write bits.