i960® VH Embedded-PCI Processor

Specification Update

November 1998

Notice: The 80960VH may contain design defects or errors known as errata. Characterized errata that may cause 80960VH's behavior to deviate from pubished specifications are documented in this specification update.

Order Number: 273174-002

Contents

Revision History

Date	Revision	Description
11/10/98	002	Added Specification Clarification #3 and Documentation Changes #1, 2 and 3.
10/13/98	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® VH Processor Developer's Manual	273173-001
i960® VH Embedded_PCI Processor Datasheet	273179-001

Nomenclature

Errata are design defects or errors. These may cause the 80690VH's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

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

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

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

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

Summary Table Of Changes

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the 80690VH 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.

Identification Information

Topside Markings

A6040-01

Device ID Registers

Device and Stepping	Processor Device ID Register (PDIDR - 1710H) (g0)	Address Translation Unit Revision ID Register (ATURID - 1208H)	i960® Core Processor Device ID (DEVICEID - FF00 8710H)
80960VH A-0	0x08864013	0x00	0x00823013

Errata

Item	Stepping			Page	Status	Errata
	A-0	#	#			
1	X			11	NoFix	Parity checking for inbound PCI address cycles is always enabled for the ATU
2	X			11	NoFix	DMA Descriptors appended to the end of a chain may not execute
3	X			11	Eval	Memory Controller Unit may assert an unexpected RAS# in certain memory configurations
4	X			15	Eval	Changing the limit register when a value exists in the corresponding base address register may prevent access to the address space
5	X			15	Eval	P_REQ# is not deasserted when a single DWORD transfer is retried
6	X			16	Fix	Bit 0 of the ATURID register is permanently set to a one
7	X			16	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
8	X			16	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
	A-0	#	#			
1	X			18	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

Specifications Clarifications

Item	Stepping			Page	Status	Specification Clarifications
	A-0	#	#			
1	X			19	Doc	Multiple reads of the Base Address Register after writing all 1's will return different values
2	X			19	Doc	When determining memory address block size, accesses to the Base Address Register must be 32-bit configuration cycles
3	X			19	Doc	Some PCI chipsets will break unaligned transactions into two LOCKED# transactions on the PCI bus. This can livelock the PCI bus if the LOCKED# transaction is directed at the Address Translation Unit

Documentation Changes

Item	Document Revision	Page	Documentation Changes
1	273173-001	20	Section 1.2.3, Messaging Unit
2	273173-001	20	Section 1.3, i960® Core Processor Features (80960VH)
3	273173-001	20	Section 17.4.5, Inbound Interrupt Mask Register - IIMR

Errata

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

Problem: The Parity Checking Enable bit (bit 06) in the Primary ATU Command Registers (local bus address 1204H) 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 signal from the 80690VH PCI bus. Use PCI masters that generate address parity in all cases.

Status: For the steppings affected see the Summary Table Of Changes.

2. 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.

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

Problem: The 80960VH MCU supports from one to four banks of DRAM. When the memory subsystem contains fewer than the maximum number of banks used in the 80960VH 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 80960VH DRAM bank. This may occur when the number of DRAM banks installed is less than the maximum number of DRAM banks used in the 80960VH design. Two examples of when this problem can occur are:

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

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

CASE 1 - Re-mapping 80960VH 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 80960VH 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 80960VH DRAM instead of FLASH/ROM

When the 80960VH initializes from 80960 local memory instead of FLASH/ROM, the 80960VH'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 80960VH DRAM

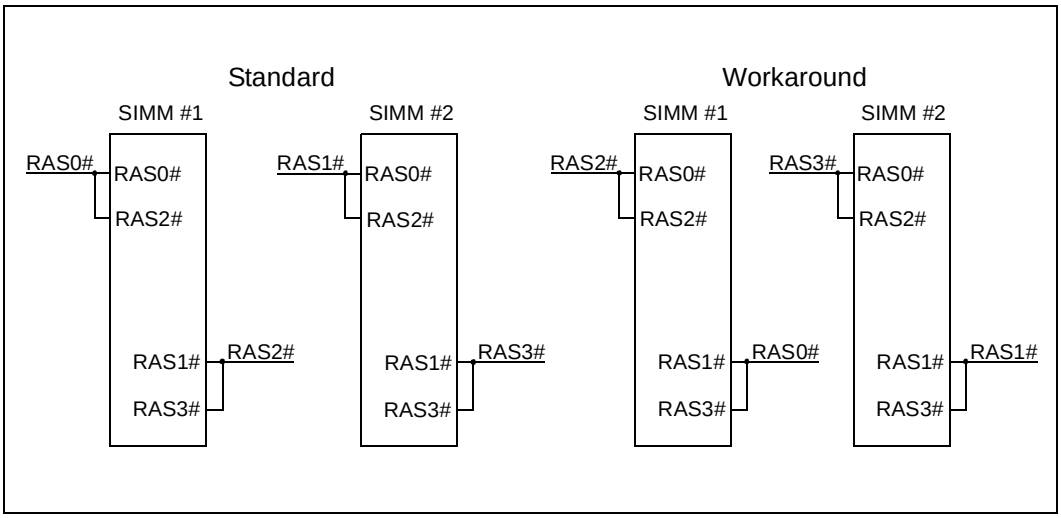
When 80960VH memory subsystem contains unpopulated DRAM banks, the DBAR must be aligned on an address boundary of a multiple of four times the DRAM bank/leaf size (for non-interleaved or interleaved memory) to ensure the correct RAS# is asserted (See Table 1). Limit 80960 local memory accesses to the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

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

CASE 2 WORKAROUND - Initializing from 80960VH 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 80960VH 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.

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

Problem: The 80960VH 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
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.

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

Problem: When the 80960VH 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 80960VH's primary PCI bus, the 80960VH 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 80960VH 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.

6. 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.

7. 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 80960VH's memory controller has a bus monitor feature which asserts LRDYRCV# if valid data is not returned for 80960 local bus accesses in 127 P_CLK 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 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 Register (PIALR) and the Inbound Translate Value Register (PIATVR) 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.

8. 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 80960VH (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 80960VH 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 80960VH 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 80960VH 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. The Memory Bank Extended MWE3:0# bits in the Memory Bank Control Register can provide one clock of address hold time during write cycles

Issue: 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 P_CLK 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.”

Specification Clarifications

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

Issue: The 80960VH provides a programmable mechanism for defining the memory block size requirements. This mechanism uses the Base Address Register (BAR) and corresponding limit register. 80960VH 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 (for example, the limit register value); all subsequent reads of the BAR will return a value other than the memory block size.

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

Issue: When determining block size requirements, the 80960VH'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.

3. Some PCI chipsets will break unaligned transactions into two LOCKED# transactions on the PCI bus. This can livelock the PCI bus if the LOCKED# transaction is directed at the Address Translation Unit

Issue: The ATU does not support PCI LOCKED# transactions. It has been observed that some PCI chipsets may split an unaligned memory read access into two LOCKED# transactions on the PCI bus. A livelock can occur if the ATU has a pending outbound write that occurs between the two LOCKED# transactions. The PCI chipset will not accept the inbound write from the ATU until its second LOCKED# read is flushed and the ATU will not accept the LOCKED# read from the PCI chipset until it completes the outbound write. Because the ATU specifically does not support PCI LOCKED# transactions, avoid performing unaligned reads of the ATU from a host processor through a PCI chipset.

Documentation Changes

1. Section 1.2.3, Messaging Unit

Issue: Page 1-2, Section 1.2.3 of Chapter 1 incorrectly reads that the MU has four messaging mechanisms.

The third sentence in Section 1.2.3 should read, “The MU has two messaging mechanisms.”

Affected Docs: *i960® VH Processor Developer’s Manual*, 273173-001

2. Section 1.3, i960® Core Processor Features (80960VH)

Issue: The core features description in the first paragraph and the Figure 1-2 label on Page 1-3, Section 1.3 of Chapter 1 are incorrect.

The first three sentences should read “The processing power of the 80960VH comes from the 80960JT processor core. The 80960JT is a new, scalar implementation of the i960 core architecture. Figure 1-2 shows a block diagram of the 80960JT core processor.” The label in Figure 1-2 should read “80960JT Core Processor Block Diagram”.

Affected Docs: *i960® VH Processor Developer’s Manual*, 273173-001

3. Section 17.4.5, Inbound Interrupt Mask Register - IIMR

Issue: The description information in Table 17-7 of Section 17.4.5 on page 17-9 is incomplete.

The description for Bit 06 in Table 17-7 should read “Reserved: must be set to ‘1’.”

Affected Docs: *i960® VH Processor Developer’s Manual*, 273173-001